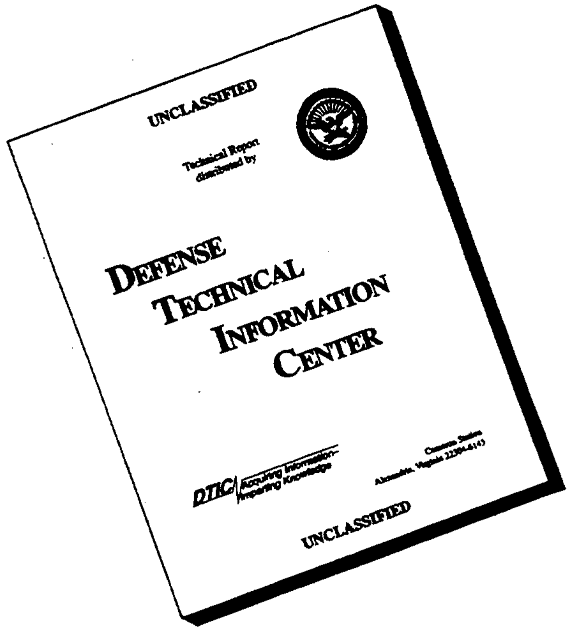
Final Report on the Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph, Volume II of II

R. Benjamin Knapp, Ph.D., Ulka Agarwal, M.S.,
Ramin Djamschidi, M.S., Shahab Layeghi, M.S.,
Mitra Dastamalchi, M.S., and Eric Jacobs, M.S.

December 1995

Department of Defense Polygraph Institute
Fort McClellan, Alabama 36205-5114
Telephone: 205-848-3803
FAX: 205-848-5332

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
December 1995

3. REPORT TYPE AND DATES COVERED
Final Report Jan 93 - Dec 95

4. TITLE AND SUBTITLE
Final Report on the Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph, Volume II of II

5. FUNDING NUMBERS
DoDPI93-P-0014

6. AUTHOR(S)
R. Benjamin Knapp, Ulka Agarwal, Ramin Djamschidi, Shahab Layeghi, Mitra Dastamalchi, Eric Jacobs

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Department of Electrical Engineering
San Jose State University
P. O. Box 720130
San Jose, California 95172-0130

8. PERFORMING ORGANIZATION REPORT NUMBER
DoDPI96-R-0002
N00014-93-I-0570

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Department of Defense
Polygraph Institute
Building 3195
Fort McClellan, AL 36205-5114

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
DoDPI93-P-0014
DoDPI96-R-0002

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Public release, distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
This project was completed to determine if fuzzy set classification could be used to accurately evaluate data collected during a psychophysiological detection of deception examination. This methodology provides an alternative to the proprietary statistical technique now commonly used. Data collected using both the Modified General Question Technique (MGQT) and the Relevant Only formats were evaluated. An extensive and, arguably, complete set of polygraph data features was identified. These polygraph data features were not individual dependent, examiner dependent, or in any way dependent on apriori or posteriori knowledge (statistics) of the data. A fuzzy K-Nearest Neighbor classifier and an adaptive fuzzy Least Mean Squares classifier were developed. A fuzzy C-Means clustering algorithm which enabled visualization of the data features was also developed. the fuzzy algorithms were "forced" to make a choice of truth versus deception; they could, however, be used to return a number that would, in near real-time, give the examiner an idea of the confidence level of the algorithm. the data were parsed such that 25% of the data were tested using an algorithm developed from the remaining 75% of the data. It is shown that only four features are needed to achieve 100% correct classification of the Relevant Only data and 97% correct classification of the MGQT data. It is suggested that any future research development, or testing or computer classification techniques, including statistical and neural techniques, include the results of this work.

14. SUBJECT TERMS
algorithm, polygraph, deception, truth, fuzzy, fuzzy logic, fuzzy set, psychophysiological detection of deception, computer

15. NUMBER OF PAGES
210

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	**PR**	- Project
G	- Grant	**TA**	- Task
PE	- Program Element	**WU**	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

- **DOD** - See DoDD 5230.24, "Distribution Statements on Technical Documents."
- **DOE** - See authorities.
- **NASA** - See Handbook NHB 2200.2.
- **NTIS** - Leave blank.

Block 12b. Distribution Code.

- **DOD** - Leave blank.
- **DOE** - Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
- **NASA** - Leave blank.
- **NTIS** - Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Report No. DoDPI96-R-0002

Final Report on the Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph, Volume II of II

R. Benjamin Knapp, Ph.D., Ulka Agarwal, M.S.,
Ramin Djamschidi, M.S., Shahab Layeghi, M.S.,
Mitra Dastamalchi, M.S., and Eric Jacobs, M.S.

December 1995

Department of Defense Polygraph Institute
Fort McClellan, Alabama 36205

Table of Contents

Volume I

Section 1: Time Domain Features for the Fuzzy Set Classification of Polygraph

Section 2: Feature Analysis of the Polygraph

Volume II

Section 3: Pattern Recognition of the Polygraph Using Fuzzy Set Theory

Section 4: Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph

Report No. DoDPI96-R-0002

Pattern Recognition of the Polygraph Using Fuzzy Set Theory

Shahab Layeghi
San Jose State University
Department of Electrical Engineering
San Jose, Ca 95106

December 1993

Department of Defense Polygraph Institute
Fort McClellan, AL 36205

Acknowledgments

I would like to express my gratitude to my advisor Dr. Ben Knapp for introducing us to this project and his precious guidance. This project was a team project which was completed by the efforts of all team members. I would like to thank my colleagues Mitra Dastmalchi and Eric Jacobs, especially Mitra whose invaluable help and support made it possible for me to complete this project. I would also like to thank undergraduate assistants Michelle Badal and Ulka Agarwal for their assistance in preparing this report.

Table of Contents

List of Figures

I. Introduction

Polygraph examinations are the most widely used method to distinguish between truth and deception. In a Polygraph examination a person is connected to a special instrument called a Polygraph which records several physiological signals such as blood pressure, Galvanic Skin Response, and respiration. The subject is asked a set of questions by an examiner. By looking at these signals the examiner is able to determine the reactions of the subject to the questions and decide whether the person was truthful or deceptive in answering each question. The problem with human classification of Polygraph tests is that the outcome depends on the examiner's experience and personal opinion. Automatic scoring of Polygraph tests has been a subject of extensive research. Several methods for Polygraph classification have been studied which are mostly based on statistical classification techniques.

In this study two main goals were presented. The first goal was finding appropriate features which have physiological basis. The second purpose was trying a new classification method based on fuzzy set theory. The advantage of using fuzzy logic is that the it does not simply assigns each input to one of the classes, but it gives the possibility of belonging of an input to each class.

Digitized Polygraph data used in this project were collected from various police stations. The data files were organized according to the test format used and were decoded to ASCII format so they can be read by Matlab. Preprocessing and feature extraction routines were implemented in the Matlab language. Three sets of files were chosen, each one of them contained 50 deceptive and 50 non-deceptive files. These files are listed in Table 10 in Appendix A. A set of features were selected based on physiological reactions, and the feature vectors for every file in each set were found. Different classification methods were studied and a Fuzzy K-nearest neighbor classifier was selected. Significance of each feature was examined according to the clustering and correct classification obtained by using that individual feature. Thirty features were selected as the final set of features and a subset of combinations of 2 to 4 of these features were examined to study the effects of combining the features on classification results. The

combination that produced the best classification for all three sets on the average was selected and the effects of changing the classifier parameters on classification was studied.

II. Polygraphs*

A polygraph examination is the most popular method used to determine if an individual is being truthful or deceptive. During an examination, a subject is asked a series of questions and the physiological responses to the questions are recorded using a polygraph. The three physical responses currently obtained from a polygraph examinations are blood pressure, respiration, and skin conductivity. Polygraph charts are usually analyzed by a human interpreter for evidence of truth or deception; however, computer algorithms are now being used to verify results [1][2].

II.1. History

The first attempt to use a scientific instrument in an effort to detect deception occurred around 1895 [3]. That was the year that Caesar Lombroso published the results of his experiments in which a hydrosphygmograph was used to measure the blood pressure-pulse changes of criminals in order to determine whether or not they were deceptive. Although the hydrosphygmograph was originally intended to be used for medical purposes, Lombroso found that it worked well for lie detection. Lombroso may have been the first to use a peak of tension test format. This was done by showing a suspect a series of photographs of children, one being the victim of sexual assault. If the suspect did not react more to the victims picture than the pictures of the other children, Lombroso concluded that the suspect did not know what the victim looked like and therefore was not the alleged perpetrator.

In 1914 Vittorio Benussi published his research on predicting deception by measuring recorded respiration tracings [4]. He found that if the length of inspiration were divide by the length of expiration, the ratio would be larger after lying than before lying and also before telling the truth than after telling the truth. In 1921 John A. Larson constructed an instrument capable of simultaneously recording blood pressure pulse and respiration during an examination [3][4]. Larson reported accurate results which prompted Leonarde Keeler to construct a better version of this instrument in 1926 [3][4].

* This section is exerpted from [17]

The use of galvanic skin response in lie detection began during the turn of the century. It's usefulness, however, did not become evident until the 1930's during which time several articles written by Father Walter G. Summers of Fordham University in New York [4]. In these articles he reports over 90 criminal cases in which examination using the galvanic skin response had all been successful and confirmed by confession or supplementary evidence. The usefulness of the galvanic skin response prompted Keeler to add an galvanometer to his polygraph. At the time of Keelers death in 1949, the Keeler Polygraph recorded blood pressure-pulse, respiration, and galvanic skin response [3].

II.2 Modern Test Formats

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format consists of an ordered combination of relevant questions about an issue, control questions that provide a physical response for comparison, and irrelevant questions that also provide a response or the lack of a response for comparison [1][4]. Three general types of test formats are in use today. These are Control Question Tests, Relevant-Irrelevant Tests, and Concealed Knowledge Tests. Each of the general test formats may have a number of more specific variations. Each test consists of two to five charts containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [3][4].

The concealed knowledge test, also called peak of tension test, is used when facts about a crime are known only by the investigators and not by the public. In this case, a subject would not know the facts unless he or she was guilty of the crime. For example, if a gun was used in a crime and the public did not know the caliber, an examiner could ask a suspect if it was a 22 caliber , a 38 caliber, or a 9 mm. If the gun used was a 9 mm and the suspect was deceptive, a polygraph chart would probably indicate evidence of deception.

A control question test is often used in criminal investigations. In this type of test a series of relevant, irrelevant, and control questions are asked. A relevant question is one which is specific to the crime being investigated. For example, " Did you steal the money?". A control question is designed to make the subject feel uncomfortable. It is not specific to the crime being investigated however it may be related in an indirect way. A control

question that could follow the relevant question stated above is "Have you ever taken anything that did not belong to you?". The control questions are compared to the relevant questions and if the responses to the relevant questions are greater, the subject is usually classified as deceptive. Irrelevant questions are used as buffers. Examples of irrelevant questions are "Are the lights in this room on?" or "Is today Monday?".

Relevant-Irrelevant tests are usually used to test people trying to obtain security clearance or get a job. In this test, relevant questions are compared to irrelevant questions. Very few control questions are asked. The purpose of control questions in this test is to make sure that the subject is capable of reacting at all.

II.3 Present Day Equipment

The most popular polygraph machines today are the Reid Polygraph developed in 1945 and the Axciton Systems computerized polygraph developed in 1989 [1][11]. The Reid polygraph scrolls a piece of paper under pens that record the biological signals. The Axciton polygraph digitizes physiological signals and uses a computer to process them. The sampling frequency of the Axciton machine is 30 Hz. Axciton provides a computer based system for ranking the subject responses but allows printouts of the charts to be scored by hand the traditional way. Both machines record the same biological signals using standard methods. Blood pressure is measured by placing a standard blood pressure cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area and the chest of the subject. This results in two signals, an upper and lower respiratory signal. Skin conductivity is measured by placing electrodes on two fingers of the same hand.

III. Feature Extraction and Classification

III.1 Introduction

The problem of Classification of Polygraph data like other pattern recognition problems can be considered of consisting of several main stages. Figure [1] shows these stages and the relationship between them. At the beginning data is preprocessed so that noise and redundancies are removed from data and feature extraction can be done more accurately. The next stage is feature extraction. In this step data is read and appropriate features are extracted from it. This is a very important step in all pattern recognition problems, because the purpose of pattern recognition is finding similarities in data that belong to the same class, and features are elements that represent these similarities. Therefore, a good set of features can lead to good classification whereas a satisfactory result cannot be achieved with an inappropriate set of features. Having a set of features, the next step is to use a method to classify data using these features. These steps as applied to Polygraph classification are described in more details in the following sections.

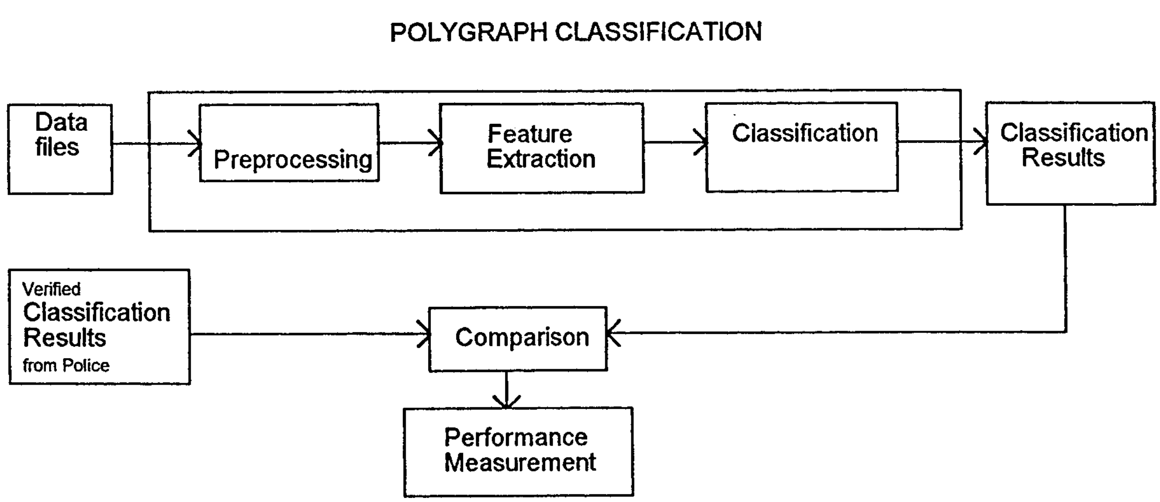


Figure 1

III.2. Preprocessing

Polygraph data consists of signals from four different channels: galvanic skin response (GSR), blood pressure, higher respiration, and lower respiration. First blood pressure signal was decomposed into a high frequency component showing heart pulse, and a low frequency component showing blood volume. Derivative of the blood volume channel was taken and used as another channel. These six derived signals were detrended and filtered. For more details on preprocessing refer to [17].

III.3. Feature Extraction

In this step appropriate features are selected and extracted. Feature extraction is itself divided into several steps. Figure [2] shows different stages involved in feature extraction.

By feature gathering we mean selecting features that might have useful information in them. Feature Combination is a special step in polygraph classification. In this step features derived for different questions in a test are combined to build a single feature. feature selection is a step in which a small number of features is selected from the main feature set to be used in final classifier section.

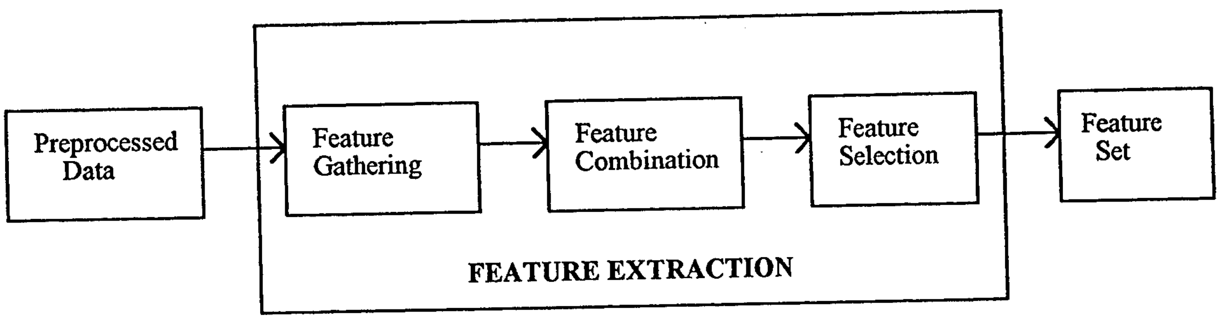


Figure 2

III.3.1. Feature Gathering

Features that possibly convey some information in them were selected and extracted in this stage. Literature about Polygraph were studied and several Polygraph examiners were interviewed to find out what had been done about this problem and what characteristics in a signal are used as indicators of truth or deception. In general features are divided into three main groups, time domain features, frequency domain features and correlation features. Time domain features are mostly standard characteristics like mean, standard deviation, median and so on. Some more specific time domain features were also added, such as the ratio between inhalation and exhalation. Auto Regressive parameters were also extracted and tried as features. To extract each feature for each question a time frame was considered that started with a specific delay after each question was asked and lasted for a specific amount of time. Different time frames were used for different channels because each channel represents a different physiological parameter. Frequency domain features include fundamental frequency, magnitude of power spectral density at fundamental frequency, coherency at fundamental frequency and so on. Figure 3 shows the feature gathering and the decisions that involved in this step.

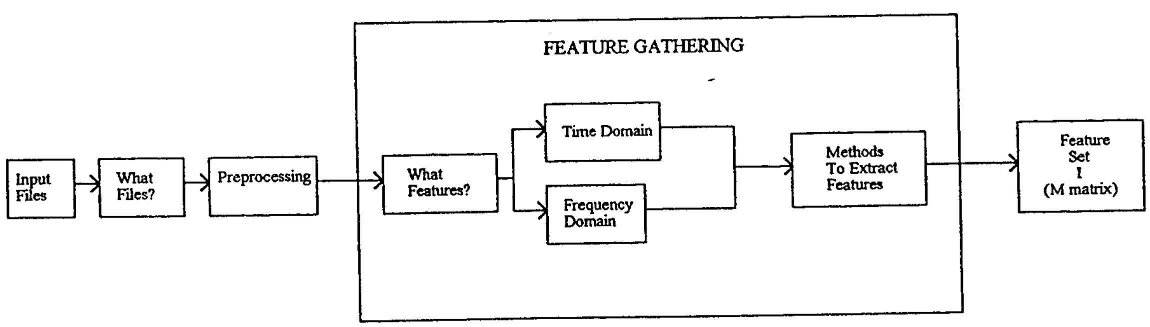


Figure. 3

For every question in a test 93 features were selected and extracted . Also 6 Integrated Spectral Density features were used which directly compare each relevant question to the nearest control question. The total number of features derived for each test was :

$$93 \times 10 + 6 \times 5 = 960$$

This was repeated for all the tests in feature sets 1, 2 and 3. The results of each set were saved in a 960x100 matrix called the M matrix.

For a detailed description of time domain features and frequency domain features refer respectively to [17] and [16].

III.3.2. Feature Combination

As mentioned earlier each feature is extracted for all questions in a test, that is for relevant, irrelevant, and control questions. In a polygraph test responses to relevant questions are compared to responses to irrelevant and control questions. But in any test there are several questions of each type and many methods can be used to combine them. Figure [4] shows different methods to combine the features. It was decided not to use irrelevant questions in this study, because in a Controlled Question Polygraph Test comparison between the responses to relevant and control questions is the most important factor. For most of the features seven methods were tried to combine features of different questions in a test. For the last six features three ways to combine them were tried. These methods were finding the average, maximum and minimum of relevant-control pairs. The first 93 features combined in seven ways and six integrated spectral density features were combined in three ways so the total number of features at this stage was equal to:

$$(93 \times 7) + (6 \times 3) = 669$$

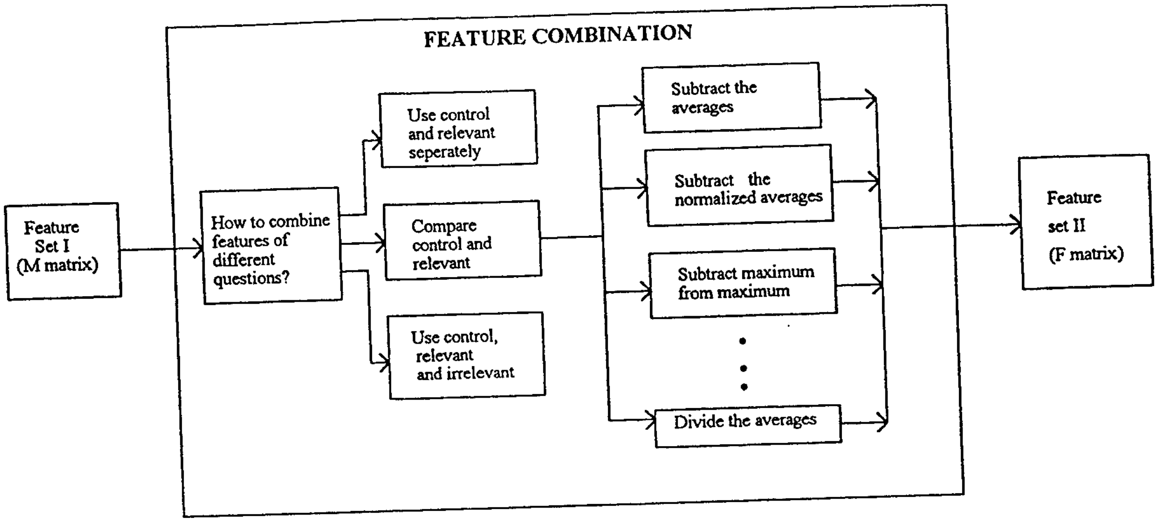


Figure 4

III.3.3 Feature Selection

Feature selection was done in two independent steps, reduction and combination. Figure [5] shows the relationship of these two steps. These two steps are explained in the following two sections.

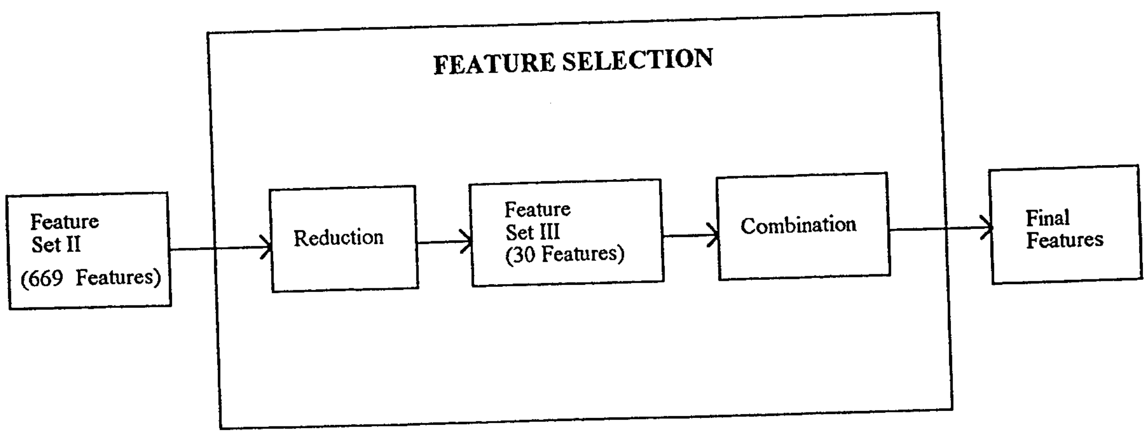


Figure. 5

III.3.3.1 Feature Selection (Reduction)

The next step in our Feature Extraction was to reduce the number of features to a number so that a practical algorithm can be used to select the feature set from them. It was decided to bring down the number of features from 669 to 30 at this step. Two different methods were chosen to test the features one at time to find the best 30. The first method was using the KNN classifier to classify the data files using one feature at a time. It was decided to use a Fuzzy version of K-nearest neighbor algorithm. The value 5 was selected for the K because it seemed that it gave better results than the other values for 1 feature classification . Also a threshold of 0.5 was used to defuzzify the output of the classifier. Refer to the section on classification for the reason of choosing this classifier. The second method was using the scatter criterion is given below.

$$J = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \qquad (1)$$

m_i = mean of class i, s_i = standard deviation of class i

This criterion measures the distance between the means of the two classes, normalized over the sum of the variances. Therefore the more compactly the samples in each class re separated, the higher will be the value of J.

The two methods were run on three sets of data. At this point a method was needed to choose the features. Different methods are possible for this step. The method that was followed is shown in figure [6] and explained below.

At first the results of KNN and scatter criterions were averaged for 3 sets of data so that features that work well for all data sets would be selected. As mentioned in an earlier section for Basic features 1 to 93, 7 features and for the features 94 to 99, 3 features were derived. Because these features are derived from one basic feature and are strongly correlated, it was decided to choose only one from them. So the best feature from these sets of 3 or 7 was selected, and the results were sorted.

Two sets of 30 features were found using the above mentioned criterions. The next step was choosing 30 features from these 60. This was done by examining the tables and selecting the features that showed a good performance in both cases or had a special physical meaning.

This set of features is the final set used for examining and selection. Table 1 in Appendix A shows these features with their corresponding meaning, channel used to derive the feature, and the method to combine the features for different questions.

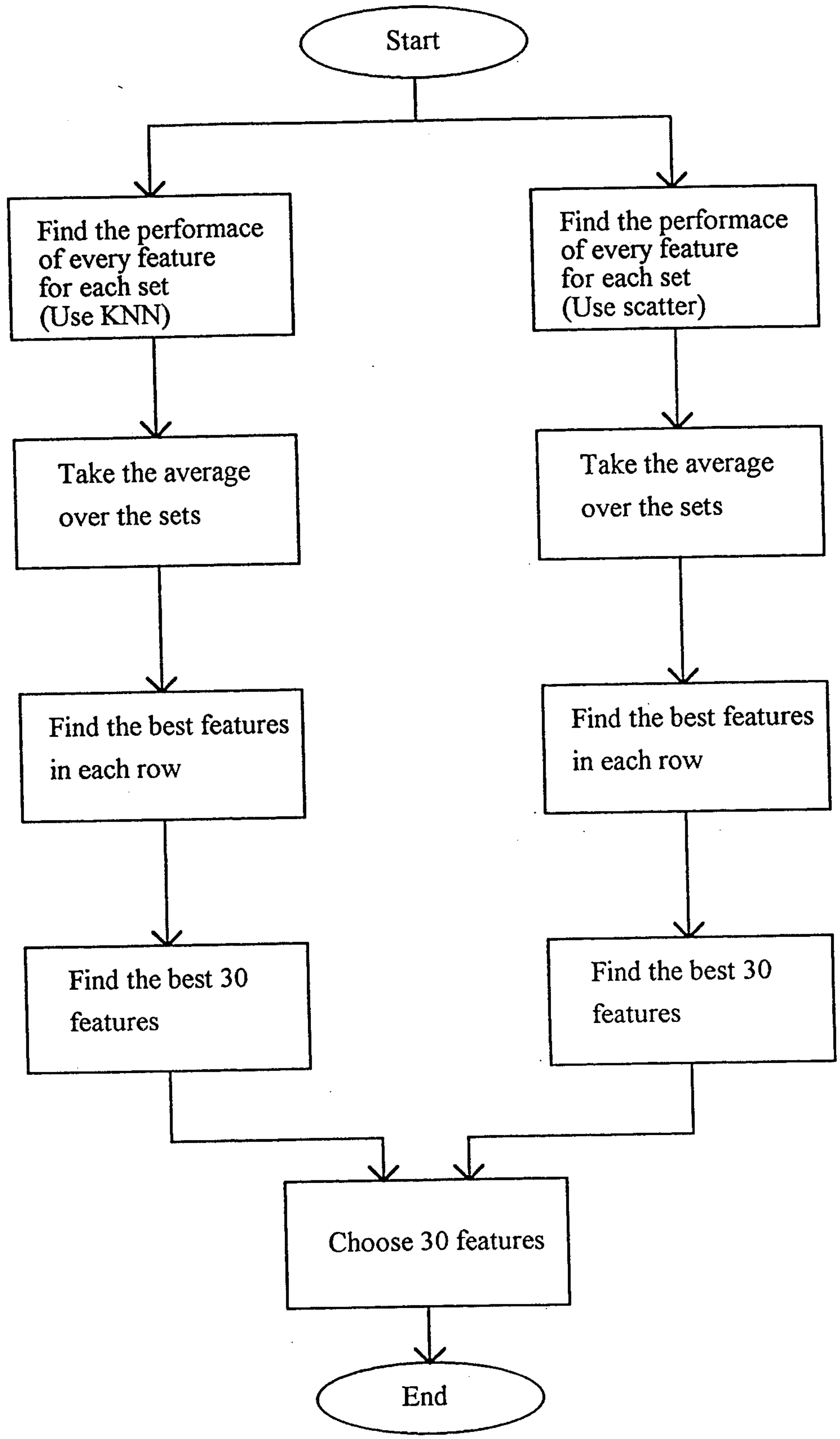


Figure. 6 Feature Selection (Reduction)

III.3.3.2 Feature Selection (Combination)

The number of features was reduced to 30 in the Feature Reduction step. This number should be further reduced because there is 100 samples in each data file, and using 30 features in a classifier might give very good results for that particular data set, but it won't be able to generalize. At this step measuring the performance of individual features is not a very logical method. Because for example features 'A' and "B' might be good features individually, but combining them might not necessarily give better results. Whereas feature 'C' that might not be a very good feature by itself might improve the classification if combined with feature 'A'.

Therefore the combinations of the features should be examined. Many methods are suggested to solve this problem. The most basic way is exhaustive search. That is trying all the combinations for these features. It is obvious that this is not practical when the number of features is not very small. For example choosing 10 or less features from a set of 30 and trying all the different combinations needs

$$\sum_{i=1}^{10} \binom{i}{30} = \sum_{i=1}^{10} \frac{30!}{i!(30-i)!} \approx 10^8$$

computations.

The method that was chosen was to start with all the combinations of two, find the best N ones among them, and use only these combinations to combine features in sets of 3. Then again find the best combinations of 3 and use them in combinations of 4 features.

This procedure is continued until satisfactory results are gained or features are not improved by increasing the number of features. Figure [7] shows the algorithm for this step.

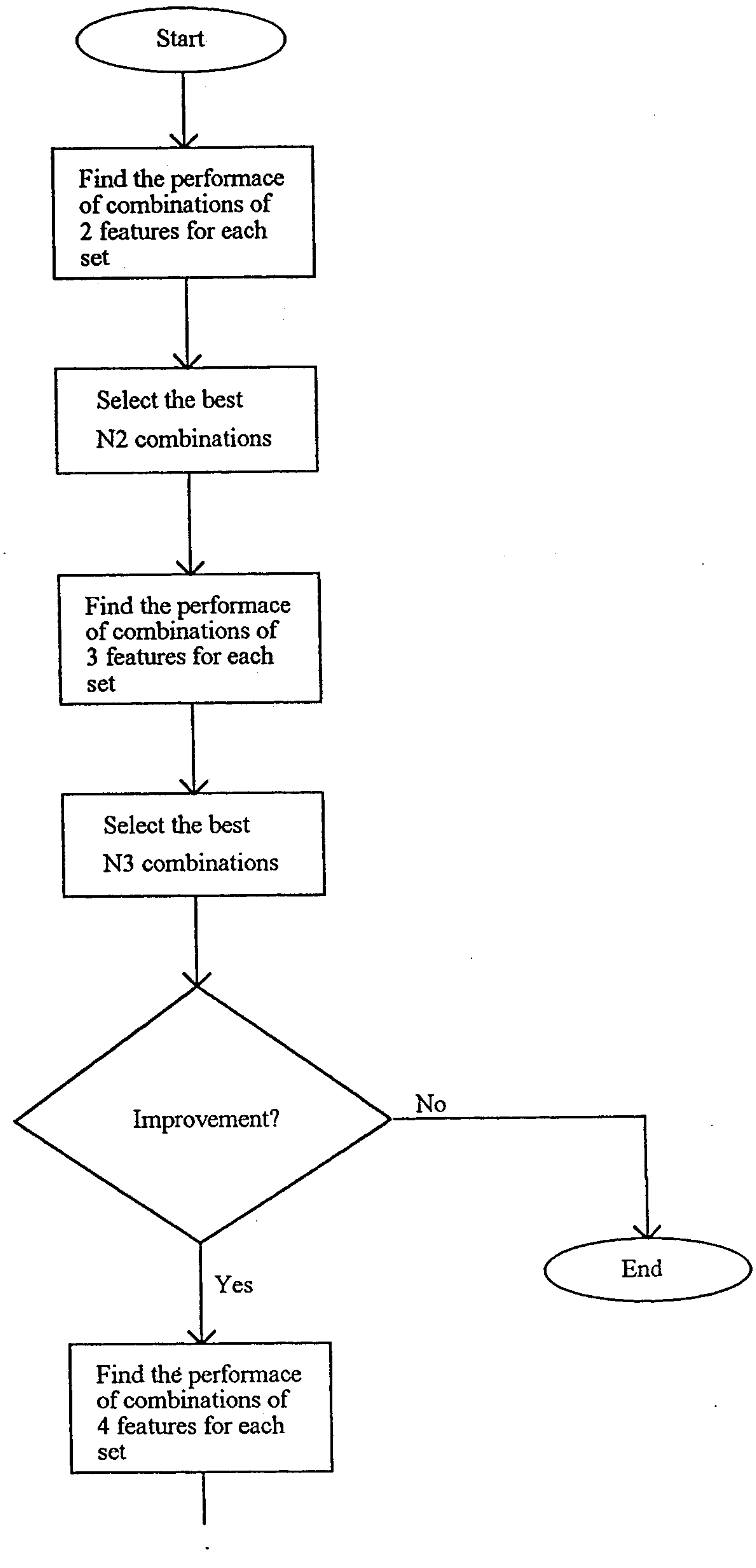


Figure 7. Feature Selection (Combination)

All pairwise combinations of the features were tried to see the classification results. The classifier used was Fuzzy K-nearest neighbor with a threshold of 0.5, and K=5. This was done for three sets of features. The results were sorted and 30 best combinations for each set were found. Also the results of classification for each combination for the 3 sets was averaged and the 30 combinations that gave best results on the average were found. These combinations are shown in Table 2 in Appendix A.

It was decided to select 20 sets of pairwise combinations to use in combinations of 3. Results for sets 1-3 and Average were studied and combinations that showed a good result in one of the sets or had a good average were selected. Table 3 in Appendix A shows these combinations.

The same steps were repeated to study the combinations of 3 and 4 features. The results are shown in Tables 4 and 6 in Appendix A. Because of time limitations it was decided not to go further from combinations of 4 features.

III.3.4 Discussion about the results:

The classification results improved consistently by increasing the number of features from one to four. The features that showed the best result for the three sets were features {5, 9, 21, 23}with 81 percent correct classification. These features represent Maximum Of GSR, Difference between Maximum and Minimum of High Cardio, Maximum of Lower Respiratory, and the Difference between Maximum and Minimum of Upper Respiratory. These features show approximately the same classification results for all three sets which is 81 percent.

Other combinations of features also gave comparable results. For example {5, 21, 23, 29} and {5, 11, 21, 23}, and {5, 10, 21, 23}. Note the repetition of {5, 21, 23}. Refer to the table 1 in Appendix A for a meaningful listing of the features. It is very notable that feature sets that show the best classification results has features that come from different channels. It can be concluded that signals from different physiological channels convey independent information, so that using features extracted from them improves the classification.

Another point to notice is that data set three shows better classification results than the two other sets, 87 percent versus 81 percent for the sets one and two. The feature set that gives the best result for data set three is {9, 14, 19, 24}. This feature set gives 87.4 percent correct classification for data set three. The feature set {5, 9, 21, 23} that gives the best classification on the average, has approximately the same results for all three sets, 81 percent. The polygraph tests that were used in this project came from several sources and were done by different examiners that used slightly different methods. Fifty consecutive tests were used to build each data set. So it is possible that some characteristic exists in the deceptive files of data set three that results in better classification. This is a matter of future investigation.

III.4. Classification

The classifier is the final stage in a pattern recognition system. The inputs to the classifier are usually a set of feature vectors. The classifier ordinarily assigns each input to one of the classes. There are many methods to design a classifier. The classifier could be designed after studying the distribution of samples of each class, or a learning classification algorithm can be implemented. We were not sure about the shape of clustering and the distribution of samples for deceptive and non deceptive classes, and it was possible that samples for one class cluster around more than one point in space. It was decided to use the K-nearest neighbor classifier* in this project because it does not explicitly use the distribution of the samples.

One of the characteristics of the conventional classification methods is that they assign each input to one of the possible classes (crisp Classification) or find probability distributions of belongingnesses of the inputs to the classes. While the way that humans think and classify objects is fundamentally different. Each object can be considered to belong to more than one class at the same time, and there are degrees of belongingness for each class. This is the basic idea that is followed in Fuzzy Logic. It was decided to follow a Fuzzy Logic based classifier in this project, because the output will be the possibility of deception and a person will not be considered completely deceptive or non deceptive.

Conventional K-nearest neighbor algorithm and a Fuzzy version of it are described in the following two sections.

---

* We are indebted to Professor R. Duda for suggesting KNN classifier.

III.4.1. K-Nearest Neighbor Algorithm

K-Nearest neighbor algorithm is a supervised classification method. There is no need for the training or adjusting the classifier. A set of labeled input samples is given to the classifier. When a new sample is given to the system, it finds its K nearest neighboring samples, and assigns this sample to the class that the majority of the neighbors belong to. K could be any positive integer. When K is set to 1, the algorithm is called the nearest neighbor algorithm. In this case each new sample is assigned to the class of its nearest neighbor. If K is greater than 1, it is possible that there is no majority class. To remove this tie, the sum of the distances of the new sample to its neighbors in each class is computed and the sample is assigned to the class that has the minimum distance. The main advantage of using this method is that the samples of each class are not needed to cluster in a pre specified shape. For example for a two class classification, the K-nearest neighbor classifier can still give very good results if the samples of each class are clustered in two distinct points in the space. The algorithm for the K nearest neighbor is shown in figure 8. It is supposed that C is the number of classes, K is the number of neighbors in KNN, $x_i\, x_i$ is the *ith* labeled sample and y is the input to be classified.

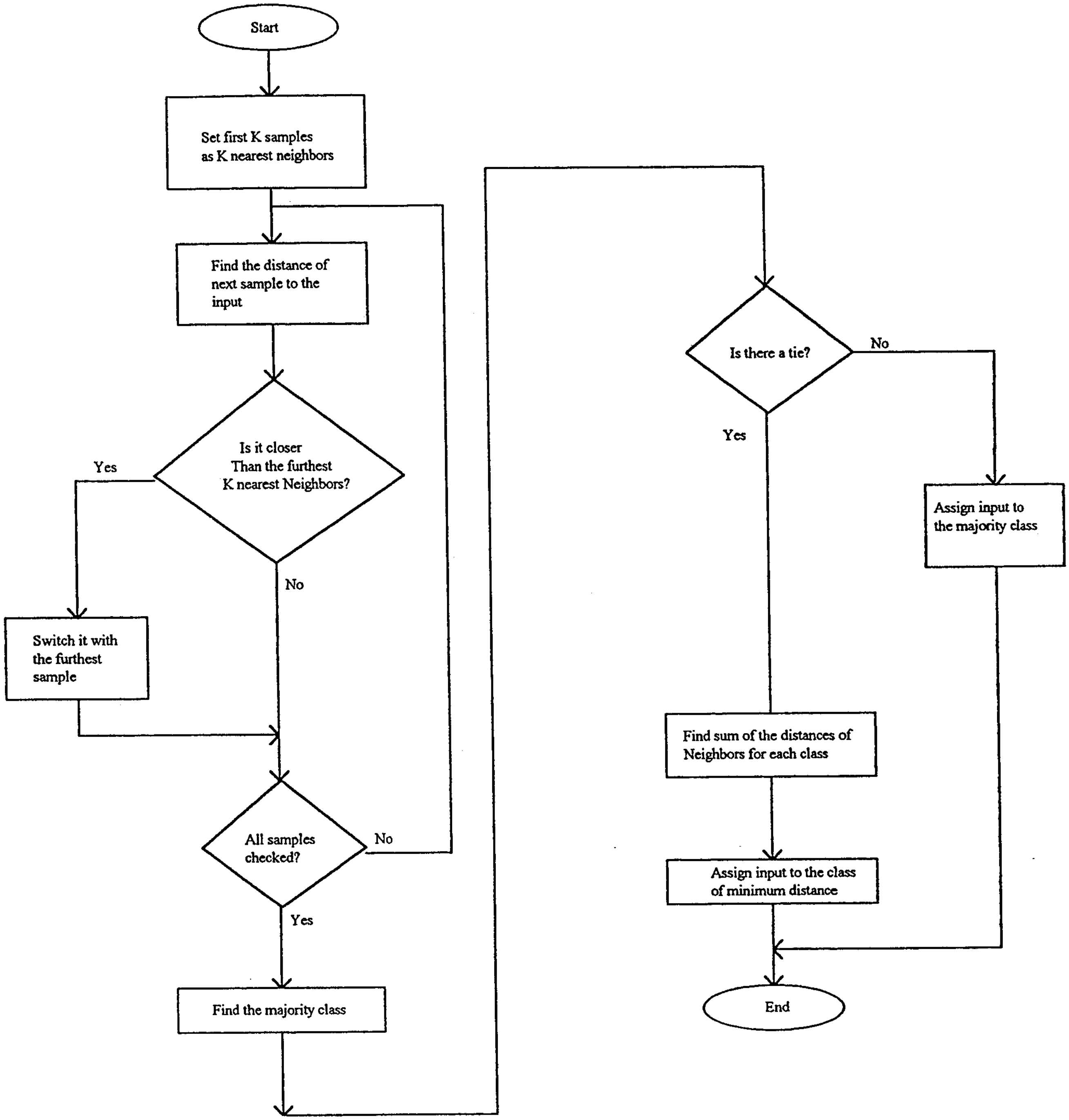


Figure 8. K Nearest Neighbor Algorithm

III.4.2. Fuzzy K Nearest Neighbor Algorithm

The fuzzy K nearest neighbor algorithm uses the same idea of conventional K nearest neighbor algorithm, that is finding the K samples that are closest to sample to be classified. But there is a conceptual difference in classification. When fuzzy classification is used, the input is not assigned to a single class. Instead, the degree of belongingness of the input to each class is determined by the classifier. By using this method more information is obtained about the input. For example if the result of classification determines membership of an input to class A is 0.9 and to class B is 0.1, it means the input belongs to class A with a very good possibility. But if the membership to class A is 0.55 and to class B is 0.45, it means that we cannot be very sure about the classification of the input. If the crisp classifier is used, in both cases the input will be assigned to class A and no further information is obtained.

Refer to [14, 15] for more detailed discussions about fuzzy K nearest neighbor algorithms. The flowchart for a fuzzy K nearest neighbor classifier is drawn in figure 9.

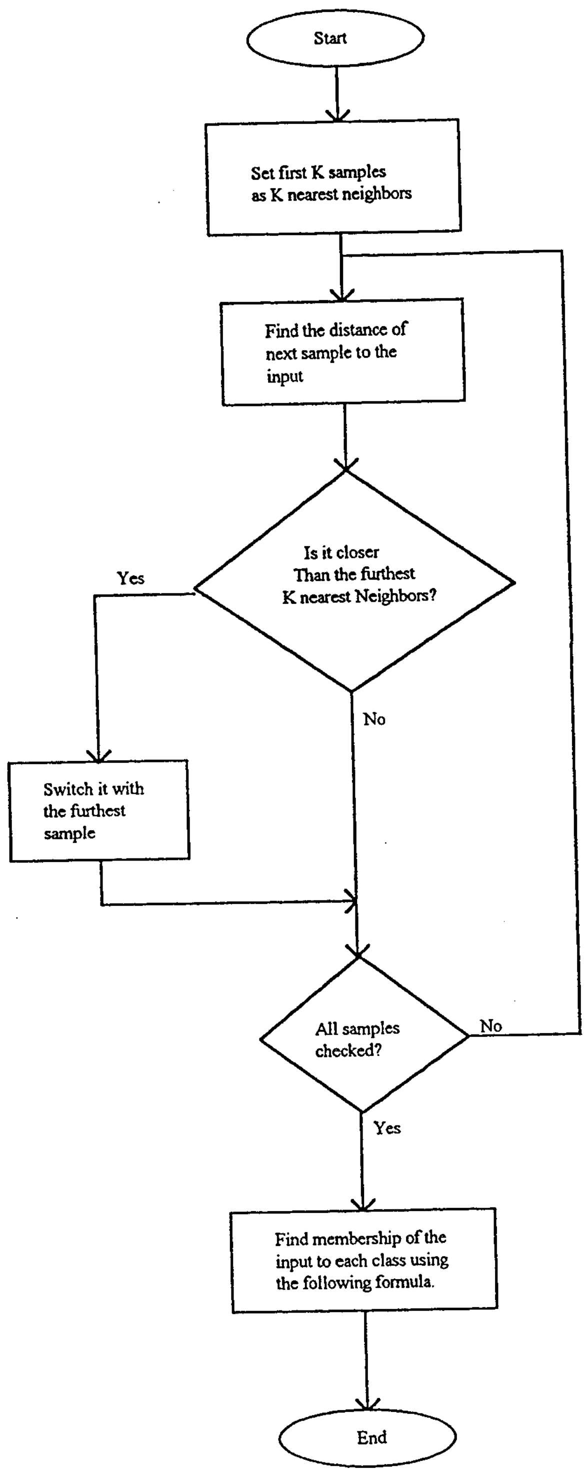
The first step in the fuzzy K nearest neighbor algorithm is the same as first step in crisp classifier. In both cases K nearest neighbors of the input are found. While in crisp classifier the majority class of the neighbors is assigned to the input, in Fuzzy classifier membership of the input to each class should be found. In order to do so the membership vector of each sample is combined to obtain the membership vector of the input. If the samples are crisply classified, membership vectors should be assigned to them. One method to do so is to assign the membership of 1 to the class that it belongs to, and membership of 0 to other classes. Other methods assign different memberships to the samples according to its distance from the mean of the class, or the distances from the nearby samples of its own class and the other classes.

When the membership vectors of the labeled samples are specified, they are combined to find the membership vector of the unknown class. This procedure should be done in a way that samples that are closer to the input have more effect on the resultant membership function. The following formula uses the inverse distance to weigh the membership

functions. x is the input to be classified, x_j is the jth nearest neighbor and u_{ij} is the membership of the jth nearest neighbor of the input in class i. D(x,y) is a distance measure between the vectors x and y which could be the Euclidean distance.

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}(1/D(x,x_j)^{\frac{1}{m-1}})}{\sum_{j=1}^{K}(1/D(x,x_j)^{\frac{1}{m-1}})}$$

m is a parameter that changes the weighing effect of the distance. When $m >> 1$, all the samples will have the same weight. When m approaches 1, the nearest samples have much more effect on the membership value of the input.



$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}\left(1/D(x,x_j)^{\frac{1}{m-1}}\right)}{\sum_{j=1}^{K}\left(1/D(x,x_j)^{\frac{1}{m-1}}\right)}$$

Figure 9. Fuzzy K-Nearest Neighbor Algorithm

III.4.3. Methods and Discussion:

As mentioned in an earlier section the classifier was needed to compare the effectiveness of single features and to choose the combinations of the features that gave the best classification results. Therefore, the classifier was selected and used before the final feature set was determined. The classifier might change the results of the classification and finding the best classifier is not a trivial task. For example using the value of 10 for K may change the set of 30 best features that was found by using K = 5.

It is not practical to try all different cases for different classifiers and different parameters of classifiers, so it was decided to use a classifier with fixed parameters up to the point that final set of features were selected. The classifier as mentioned earlier was a Fuzzy K-nearest neighbor with the following parameters:

K = 5,

m = 2,

Defuzzification threshold = 0.5;

It should be noted that in order to save computation time throughout this project, each set of files was randomly broken into a training and a testing set. Each file in the testing set was classified using the labeled files in training set. Each experiment was repeated 20 times, and the results were averaged. The number of files that were used for training and testing were accordingly 75 and 25. In the last stage of experiments after the final feature set had been fixed, instead of randomly selecting testing and training files, one file was kept for testing each time and the experiment was repeated 100 times changing the test file.

After the final feature set was selected (Refer to the section on Feature Extraction), different values for K were tried on fuzzy and crisp classifier to compare the two classifiers and find the best parameters. In addition to percentage of correct classification a measure of performance was also used which is explained below.

The measure that is used to compare the performance of fuzzy classifier is the root mean square of the distances between the output of the classifier and the correct class. The correct ouput of the classifer should be 0 for non-deceptive cases and 1 for the deceptive

ones. For example if for a deceptive sample the classifier output is 0.8, 0.2 is the distance between the output and the correct class. The same measure is used for the crisp classifier. In the case of the crisp classifier the distance is always 0 for correct classification and 1 for incorrect classification.

For the fuzzy classifier the threshold used for defuzzification was also changed to find the optimum value. Tables 7 and 8 in Appendix A show the results. The best classification on the average over three sets is obtained using the fuzzy classifier with K = 6, and threshold = 0.6 . Using this values correct classification of 81.6 percent was achieved. The best result using the crisp classifier was 80.6 percent which was obtained using K=6. The performance measures for the fuzzy and crisp classifiers were accordingly 0.3915 and 0.4377 which shows fuzzy classifier has a better performance in this respect.

One final experiment that was done is explained below. In a Polygraph examination a set of questions is repeated one to five times and the decision is made by considering the responses to all these charts. In this project each chart was classified separately. As the final experiment responses to all the charts in a Polygraph examination were combined and classified as deceptive or non-deceptive. The way they were combined was finding the majority class and assigning the case to that class. In the case that equal number of files classified as deceptive and non-deceptive, the membership function of the files was averaged and the case was classified according to this value. The classification results for all the files in sets 1 to 3 are shown in Table 9 in Appendix A. The number of cases in each set was 35. The number of misclassified cases in sets 1 to 3 are 5, 7, and 3, which correspond to correct classifications of 85.7, 80.0, and 91.4 percent.

IV. Conclusion and future work

The set of four features that showed best classification results in this project were Maximum of GSR, Upper Respiration and Lower respiration signals, and the difference between the Maximum and Minimum of High Cardio signal. These are all very simple time domain features. The best classification was obtained using the fuzzy classifier with K = 6, and threshold = 0.6 . Using this values correct classification of 81.6 percent was achieved. By combining all the files in a Polygraph examination 85.7 percent correct classification was achieved on the average.

There are several suggestions for the future work. First is to repeat this work with larger sets of data files and observe the generalizability of the feature sets obtained in this research. A possible way to improve the results is to change time frames used to extract each feature for every question. In this way the optimum time for obtaining a response could be found. Another suggestion is to try different methods for fuzzification and defuzzification of feature vectors to optimize the fuzzy classifier.

REFERENCES

[1] Dale E. Olsen, et. al., "Recent developments in polygraph testing: A research review and evaluation - A technical memorandum, " Washington DC: US Government Printing Office 1983.

[2] John C. Kircher and David C. Raskin, "Human versus computerized evaluations of polygraph data in a laboratory setting, " Journal of Applied Psycology, Vol.73, 1988 No 2, pp. 291-308

[3] John E. Reid and Fred E. Inbau, Truth and Deception: The Polygraph (Lie Detector) Technique, The Williams & Wilkins Company, Baltimore, Md., 1966

[4] Michael H. Capps and Norman Ansley, "Numerical Scoring of Polygraph Charts: What Examiners Really Do", Polygraph, 1992, 21, 264-320

[5] L. A. Zadeh, "Fuzzy sets", Information and Control, vol. 8, pp. 338-332, 1965

[6] James C. Bezdek and Sankar K. Pal, Fuzzy Models for Pattern Recognition Methods that Search for Structures in Data, IEEE Press, Piscataway, NJ. 1992

[7] L. A. Zadeh, "Calculus of fuzzy restrictions," in: L. A. Zadeh, K. S. Fu, K. Tanaka and M. Shimura, eds., Fuzzy Sets and Their Applications to Cognitive and Decision Processes, Academic Press, New York, 1975, pp. 1-39

[8] Bart Kosko, Neural Networks and Fuzzy Systems, New Jersey : Prentice-Hall, Inc., 1992.

[9] Brian M. Duston, " Statistical Techniques for Classifying Polygraph Data ", Draft, November 24, 1992

[10] Howard W. Timm, " Analyzing Deception From Respiration Patterns " , Journal of Police Science and Administration, 1982, 1, 47 - 51.

[11] Personal communication with Richard Petty (polygraph examiner), June 1993

[12] Personal communication with Christopher B. Pounds (University of Washington), May 1993

[13] Personal communication with Howard Timm, May 1993

[14] J.M. Keller, M.R. Gray and J.A. Givens, "A Fuzzy K Nearest Neighbor Algorithm", IEEE Trans. on Syst. Man. Cybernetics, vol SMC-15, no. 4

[15] J.C. Bezdek and Siew K. Chuah, "Generalized K-Nearest Neighbor Rules, Fuzzy Sets and Systems vol. 18 (1986)

[16] Mitra Dastmalchi, "Feature Analysis of the Polygraph", Master's Project, San Jose State University, December 1993

[17] Eric Jacobs, "Time Domain Feature Extraction of the Polygraph", Master's Project, San Jose State University, December 1993

Appendices

Appendix A:
Tables

No.	feature	Description	Channel	Method
1	10mean	mean	GSR	1
2	10curve	curve length	GSR	2
3	10med_dif	median of the derivative	GSR	1
4	10max_min	minimum subtracted from the maximum	GSR	2
5	10max	maximum of the signal	GSR	1
6	10mdif	mean of derivative	GSR	3
7	20curve	curve length	High Cardio	1
8	20ampcard	amplitude of the peaks	High Cardio	1
9	20max_min	minimum subtracted from the maximum	High Cardio	4
10	20max	maximum of the signal	High Cardio	4
11	20min	minimum of the signal	High Cardio	1
12	30med_dif	median of the derivative	Low Cardio	3
13	30max	maximum of the signal	Low Cardio	1
14	40mean	mean	Derivative of Low Cardio	1
15	40max	maximum of the signal	Derivative of Low Cardio	1
16	50curve	curve length	Lower Respiratory	6
17	50ampr	amplitude of the peaks	Lower Respiratory	2
18	50peaknumr	number of the peaks	Lower Respiratory	5
19	50ie	inhalation divided by exhalation	Lower Respiratory	5
20	50max_min	minimum subtracted from the maximum	Lower Respiratory	2
21	50max	maximum of the signal	Lower Respiratory	6
22	60max_min	minimum subtracted from the maximum	Upper Respiratory	2
23	60max	maximum	Upper Respiratory	3
24	10std	standard deviation	GSR	2
25	20std	standard deviation	High Cardio	1
26	50std	standard deviation	Upper Respiratory	6
27	20armod1	auto regressive parameter	High Cardio	7
28	26psdcoh1	max cross spectral density	High Cardio, Lower Respiratory	1
29	10isd1	frequency of maximum integrated spectral difference of control-relevant pair	GSR	1*
30	20isd1	area under integrated spectral difference	High Cardio	3*

Methods: 1=Difference of Averages, 2=Normalized Average, 3=Max-Max, 4=Min-Min, 5=Max-Min, 6=Min-Max, 7=Max/Min , 1*=Average of relevant-control pairs, 3*=Max of relevant-control pair.

Table 1. Selected Features

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2
74.2000	8.0000	18.0000
74.0000	10.0000	21.0000
73.0000	5.0000	7.0000
72.0000	24.0000	26.0000
71.8000	23.0000	24.0000
71.6000	4.0000	26.0000
70.4000	25.0000	26.0000
70.4000	18.0000	25.0000
70.2000	24.0000	27.0000
70.2000	9.0000	21.0000
70.0000	5.0000	27.0000
69.6000	11.0000	21.0000
69.6000	9.0000	24.0000
69.4000	11.0000	27.0000
69.4000	5.0000	26.0000
69.2000	8.0000	19.0000
69.2000	5.0000	18.0000
69.0000	25.0000	27.0000
69.0000	9.0000	18.0000
69.0000	5.0000	23.0000
68.8000	24.0000	30.0000
68.8000	18.0000	20.0000
68.8000	17.0000	20.0000
68.8000	4.0000	15.0000
68.6000	22.0000	24.0000
68.4000	6.0000	24.0000
68.4000	1.0000	27.0000
68.2000	15.0000	24.0000
68.2000	9.0000	26.0000
68.2000	5.0000	19.0000

Table [2.1] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2
74.4000	5.0000	23.0000
74.4000	4.0000	27.0000
74.2000	4.0000	15.0000
74.0000	20.0000	24.0000
73.6000	16.0000	24.0000
73.2000	3.0000	27.0000
72.8000	27.0000	30.0000
72.6000	4.0000	30.0000
72.6000	4.0000	7.0000
72.4000	5.0000	25.0000
72.2000	24.0000	30.0000
72.2000	8.0000	27.0000
72.2000	4.0000	17.0000
72.2000	4.0000	16.0000
72.0000	24.0000	27.0000
72.0000	24.0000	25.0000
72.0000	4.0000	20.0000
71.8000	7.0000	23.0000
71.8000	4.0000	10.0000
71.2000	25.0000	27.0000
70.8000	24.0000	26.0000
70.8000	8.0000	22.0000
70.6000	7.0000	27.0000
70.6000	6.0000	27.0000
70.4000	14.0000	21.0000
70.4000	14.0000	20.0000
70.4000	4.0000	8.0000
70.2000	4.0000	24.0000
70.0000	22.0000	27.0000
70.0000	17.0000	24.0000

Table [2.2] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2
81.0000	1.0000	10.0000
80.6000	9.0000	24.0000
80.4000	10.0000	24.0000
80.4000	4.0000	25.0000
80.2000	4.0000	9.0000
79.8000	5.0000	11.0000
79.2000	17.0000	24.0000
79.2000	1.0000	21.0000
79.2000	1.0000	8.0000
79.0000	1.0000	24.0000
79.0000	1.0000	11.0000
78.8000	4.0000	11.0000
78.6000	4.0000	17.0000
78.2000	24.0000	25.0000
78.2000	1.0000	14.0000
78.0000	1.0000	23.0000
78.0000	1.0000	20.0000
77.8000	23.0000	24.0000
77.8000	1.0000	5.0000
77.6000	19.0000	24.0000
77.4000	11.0000	24.0000
77.4000	5.0000	18.0000
77.2000	4.0000	19.0000
77.0000	4.0000	18.0000
76.8000	4.0000	15.0000
76.6000	5.0000	13.0000
76.6000	4.0000	24.0000
76.2000	4.0000	5.0000
76.2000	1.0000	26.0000

Table [2.3] Results of pairwise combinations of features

Percentage of correct classification for 30 best combinations in average

Percent correct	Feature 1	Feature 2
73.2667	4.0000	15.0000
72.8000	24.0000	26.0000
72.6667	4.0000	17.0000
72.6000	5.0000	23.0000
72.2667	23.0000	24.0000
72.0667	24.0000	30.0000
71.9333	20.0000	24.0000
71.8667	24.0000	27.0000
71.4667	24.0000	25.0000
71.4000	4.0000	26.0000
71.0667	4.0000	10.0000
70.9333	1.0000	8.0000
70.9333	4.0000	23.0000
70.6000	5.0000	11.0000
70.6000	4.0000	24.0000
70.5333	9.0000	24.0000
70.4667	6.0000	24.0000
70.4667	4.0000	25.0000
70.4667	4.0000	19.0000
70.4000	4.0000	30.0000
70.3333	1.0000	23.0000
70.0667	17.0000	24.0000
70.0667	1.0000	24.0000
70.0000	16.0000	24.0000
69.9333	4.0000	9.0000
69.8667	4.0000	20.0000
69.8667	5.0000	7.0000
69.8667	4.0000	7.0000
69.8000	15.0000	24.0000
69.8000	1.0000	21.0000

Table [2.4] Results of pairwise combinations of features

4	15
24	26
4	17
5	3
23	24
24	30
20	24
24	27
24	25
4	26
1	10
9	24
10	24
5	11
17	24
4	27
16	24
8	18
10	21
5	7

Table [3]. 20 combinations of 2 features selected to combine in sets of 3

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2	Feature 3
79.4000	10.0000	21.0000	26.0000
77.6000	5.0000	7.0000	23.0000
77.6000	5.0000	23.0000	11.0000
77.4000	5.0000	23.0000	21.0000
76.4000	16.0000	24.0000	18.0000
76.4000	5.0000	23.0000	19.0000
75.8000	23.0000	24.0000	19.0000
75.8000	23.0000	24.0000	15.0000
75.8000	5.0000	23.0000	7.0000
75.6000	5.0000	7.0000	22.0000
75.6000	5.0000	7.0000	21.0000
75.6000	5.0000	7.0000	16.0000
75.4000	5.0000	7.0000	14.0000
75.4000	5.0000	11.0000	10.0000
75.2000	10.0000	21.0000	19.0000
75.2000	8.0000	18.0000	6.0000
75.2000	5.0000	23.0000	2.0000
75.0000	10.0000	21.0000	16.0000
75.0000	10.0000	21.0000	8.0000
75.0000	5.0000	11.0000	18.0000
75.0000	4.0000	26.0000	14.0000
75.0000	5.0000	23.0000	29.0000
75.0000	5.0000	23.0000	25.0000
74.8000	10.0000	21.0000	9.0000
74.6000	10.0000	21.0000	12.0000
74.6000	5.0000	11.0000	23.0000
74.6000	10.0000	24.0000	9.0000
74.6000	5.0000	23.0000	10.0000
74.6000	5.0000	23.0000	9.0000
74.4000	5.0000	7.0000	19.0000

Table [4.1] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2	Feature 3
79.8000	20.0000	24.0000	12.0000
78.6000	24.0000	30.0000	19.0000
78.6000	4.0000	15.0000	28.0000
78.0000	24.0000	27.0000	19.0000
77.8000	4.0000	17.0000	19.0000
77.6000	8.0000	18.0000	4.0000
77.4000	4.0000	27.0000	19.0000
77.4000	5.0000	23.0000	21.0000
77.2000	5.0000	23.0000	29.0000
77.2000	4.0000	15.0000	27.0000
77.0000	4.0000	27.0000	18.0000
77.0000	4.0000	15.0000	21.0000
76.6000	5.0000	7.0000	23.0000
76.6000	20.0000	24.0000	3.0000
76.4000	16.0000	24.0000	30.0000
76.4000	4.0000	27.0000	25.0000
76.4000	24.0000	27.0000	10.0000
76.4000	23.0000	24.0000	30.0000
76.2000	5.0000	23.0000	3.0000
76.2000	4.0000	17.0000	2.0000
76.2000	4.0000	15.0000	26.0000
75.8000	5.0000	7.0000	15.0000
75.8000	24.0000	30.0000	4.0000
75.8000	5.0000	23.0000	28.0000
75.6000	4.0000	27.0000	15.0000
75.6000	24.0000	27.0000	26.0000
75.6000	24.0000	27.0000	1.0000
75.6000	20.0000	24.0000	25.0000
75.6000	24.0000	30.0000	16.0000
75.4000	4.0000	15.0000	8.0000

Table [4.2] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2	Feature 3
85.2000	9.0000	24.0000	19.0000
85.0000	9.0000	24.0000	22.0000
84.2000	16.0000	24.0000	19.0000
84.0000	17.0000	24.0000	9.0000
84.0000	4.0000	26.0000	17.0000
83.6000	4.0000	26.0000	11.0000
83.6000	4.0000	17.0000	9.0000
83.6000	24.0000	26.0000	17.0000
83.6000	4.0000	15.0000	9.0000
83.4000	5.0000	11.0000	24.0000
83.4000	9.0000	24.0000	21.0000
83.4000	9.0000	24.0000	17.0000
83.4000	9.0000	24.0000	14.0000
83.4000	4.0000	26.0000	9.0000
83.2000	16.0000	24.0000	1.0000
83.2000	4.0000	17.0000	26.0000
83.2000	24.0000	26.0000	9.0000
83.0000	9.0000	24.0000	12.0000
83.0000	9.0000	24.0000	6.0000
83.0000	4.0000	17.0000	11.0000
82.8000	9.0000	24.0000	18.0000
82.8000	23.0000	24.0000	1.0000
82.8000	4.0000	17.0000	24.0000
82.8000	4.0000	17.0000	8.0000
82.6000	17.0000	24.0000	19.0000
82.4000	17.0000	24.0000	8.0000
82.4000	9.0000	24.0000	2.0000
82.4000	5.0000	23.0000	29.0000
82.2000	5.0000	23.0000	10.0000
82.0000	9.0000	24.0000	26.0000

Table [4.3] Results of combinations of 3 features

Percentage of correct classification for 30 best combinations on average

Percent correct	Feature 1	Feature 2	Feature 3
78.2000	5.0000	23.0000	29.0000
77.6000	5.0000	7.0000	23.0000
77.3333	5.0000	23.0000	21.0000
76.6000	5.0000	23.0000	10.0000
76.0000	23.0000	24.0000	15.0000
75.8667	5.0000	7.0000	21.0000
75.8667	5.0000	23.0000	7.0000
75.6667	5.0000	23.0000	11.0000
75.6000	8.0000	18.0000	4.0000
75.5333	4.0000	17.0000	19.0000
75.5333	5.0000	11.0000	17.0000
75.5333	24.0000	26.0000	14.0000
75.4667	5.0000	23.0000	28.0000
75.4667	4.0000	15.0000	26.0000
75.3333	17.0000	24.0000	19.0000
75.3333	5.0000	23.0000	25.0000
75.2000	5.0000	7.0000	17.0000
75.2000	4.0000	15.0000	23.0000
75.0000	5.0000	23.0000	17.0000
74.9333	5.0000	23.0000	3.0000
74.8667	4.0000	26.0000	15.0000
74.8000	23.0000	24.0000	19.0000
74.8000	5.0000	23.0000	14.0000
74.8000	5.0000	23.0000	1.0000
74.8000	24.0000	26.0000	25.0000
74.7333	24.0000	30.0000	19.0000
74.7333	5.0000	23.0000	19.0000
74.7333	5.0000	23.0000	9.0000
74.6667	5.0000	7.0000	22.0000
74.6667	4.0000	26.0000	19.0000

Table [4.4] Results of combinations of 3 features

4	17	26
5	23	29
9	19	24
4	5	9
5	10	23
5	21	23
4	8	18
19	24	30
5	7	23
19	23	24
9	14	24
4	15	28
5	11	17
4	19	17
5	23	24
5	7	21
5	11	23
14	24	26
10	21	26
4	11	26

Table [5]. 20 combinations of 3 features selected to combine in sets of 4

Percentage of correct classification for 30 best combinations in set 1

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0000	5.0000	21.0000	23.0000	9.0000
80.6000	5.0000	7.0000	23.0000	6.0000
80.2000	5.0000	21.0000	23.0000	11.0000
79.6000	5.0000	21.0000	23.0000	10.0000
79.4000	5.0000	7.0000	23.0000	12.0000
79.4000	5.0000	10.0000	23.0000	21.0000
79.0000	5.0000	7.0000	23.0000	28.0000
79.0000	5.0000	7.0000	23.0000	19.0000
79.0000	5.0000	21.0000	23.0000	26.0000
78.8000	5.0000	11.0000	23.0000	7.0000
78.6000	5.0000	21.0000	23.0000	12.0000
78.4000	5.0000	21.0000	23.0000	15.0000
78.4000	5.0000	10.0000	23.0000	8.0000
78.0000	5.0000	11.0000	23.0000	21.0000
78.0000	5.0000	7.0000	23.0000	20.0000
78.0000	5.0000	7.0000	23.0000	14.0000
77.8000	5.0000	7.0000	23.0000	2.0000
77.8000	5.0000	21.0000	23.0000	28.0000
77.8000	5.0000	21.0000	23.0000	6.0000
77.8000	5.0000	21.0000	23.0000	3.0000
77.8000	5.0000	23.0000	29.0000	26.0000
77.8000	5.0000	23.0000	29.0000	22.0000
77.6000	10.0000	21.0000	26.0000	2.0000
77.6000	5.0000	7.0000	23.0000	22.0000
77.6000	5.0000	10.0000	23.0000	19.0000
77.6000	5.0000	23.0000	29.0000	19.0000
77.6000	5.0000	23.0000	29.0000	1.0000
77.4000	10.0000	21.0000	26.0000	9.0000
77.4000	5.0000	11.0000	23.0000	10.0000
77.4000	5.0000	11.0000	23.0000	8.0000

Table [6.1] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations in set 2

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0000	5.0000	23.0000	29.0000	14.0000
79.8000	5.0000	10.0000	23.0000	21.0000
79.6000	5.0000	21.0000	23.0000	11.0000
79.4000	14.0000	24.0000	26.0000	19.0000
79.4000	5.0000	21.0000	23.0000	9.0000
79.2000	5.0000	21.0000	23.0000	13.0000
79.0000	5.0000	11.0000	23.0000	3.0000
79.0000	5.0000	23.0000	29.0000	21.0000
78.8000	5.0000	23.0000	29.0000	6.0000
78.6000	4.0000	19.0000	17.0000	25.0000
78.6000	5.0000	21.0000	23.0000	10.0000
78.4000	4.0000	19.0000	17.0000	6.0000
78.4000	5.0000	23.0000	29.0000	19.0000
78.2000	5.0000	11.0000	23.0000	25.0000
78.2000	5.0000	11.0000	23.0000	6.0000
78.2000	4.0000	15.0000	28.0000	27.0000
78.2000	5.0000	7.0000	23.0000	11.0000
78.2000	19.0000	24.0000	30.0000	11.0000
78.0000	5.0000	21.0000	23.0000	27.0000
77.8000	19.0000	24.0000	30.0000	23.0000
77.8000	19.0000	24.0000	30.0000	16.0000
77.8000	5.0000	10.0000	23.0000	11.0000
77.6000	4.0000	19.0000	17.0000	3.0000
77.6000	5.0000	7.0000	23.0000	28.0000
77.4000	14.0000	24.0000	26.0000	20.0000
77.4000	5.0000	21.0000	23.0000	30.0000
77.2000	5.0000	11.0000	23.0000	8.0000
77.2000	4.0000	19.0000	17.0000	11.0000
77.2000	5.0000	7.0000	23.0000	26.0000
77.2000	5.0000	21.0000	23.0000	12.0000

Table [6.2] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations in set 3

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
87.4000	9.0000	19.0000	24.0000	14.0000
87.2000	9.0000	14.0000	24.0000	19.0000
87.0000	9.0000	19.0000	24.0000	11.0000
86.8000	9.0000	19.0000	24.0000	18.0000
86.6000	5.0000	21.0000	23.0000	29.0000
86.6000	9.0000	19.0000	24.0000	16.0000
86.4000	9.0000	19.0000	24.0000	21.0000
86.4000	4.0000	17.0000	26.0000	18.0000
86.2000	4.0000	11.0000	26.0000	24.0000
86.2000	4.0000	8.0000	18.0000	9.0000
86.2000	9.0000	19.0000	24.0000	22.0000
86.2000	9.0000	19.0000	24.0000	6.0000
86.0000	9.0000	19.0000	24.0000	12.0000
86.0000	9.0000	19.0000	24.0000	10.0000
85.8000	9.0000	19.0000	24.0000	26.0000
85.8000	4.0000	17.0000	26.0000	9.0000
85.6000	5.0000	7.0000	21.0000	16.0000
85.6000	5.0000	7.0000	21.0000	8.0000
85.6000	9.0000	19.0000	24.0000	8.0000
85.6000	9.0000	19.0000	24.0000	5.0000
85.6000	9.0000	19.0000	24.0000	1.0000
85.4000	9.0000	14.0000	24.0000	4.0000
85.4000	5.0000	21.0000	23.0000	1.0000
85.2000	4.0000	19.0000	17.0000	10.0000
85.2000	9.0000	19.0000	24.0000	4.0000
85.0000	5.0000	11.0000	17.0000	4.0000
85.0000	9.0000	19.0000	24.0000	2.0000
85.0000	4.0000	17.0000	26.0000	8.0000
84.8000	4.0000	11.0000	26.0000	9.0000
84.8000	5.0000	21.0000	23.0000	22.0000

Table [6.3] Results of combinations of 4 features

Percentage of correct classification for 30 best combinations on average

Percent correct	Feature 1	Feature 2	Feature 3	Feature 4
81.0667	5.0000	21.0000	23.0000	9.0000
79.9333	5.0000	23.0000	29.0000	21.0000
79.8667	5.0000	21.0000	23.0000	11.0000
79.6000	5.0000	10.0000	23.0000	21.0000
79.2667	5.0000	23.0000	29.0000	19.0000
79.1333	5.0000	21.0000	23.0000	10.0000
79.0667	5.0000	23.0000	29.0000	14.0000
79.0000	14.0000	24.0000	26.0000	19.0000
78.9333	5.0000	7.0000	23.0000	12.0000
78.8667	5.0000	21.0000	23.0000	22.0000
78.8667	5.0000	7.0000	23.0000	28.0000
78.7333	5.0000	7.0000	23.0000	6.0000
78.6667	5.0000	21.0000	23.0000	7.0000
78.5333	5.0000	21.0000	23.0000	1.0000
78.4667	5.0000	23.0000	29.0000	1.0000
78.4000	5.0000	7.0000	21.0000	8.0000
78.4000	5.0000	7.0000	23.0000	26.0000
78.2667	5.0000	7.0000	23.0000	11.0000
78.2000	5.0000	7.0000	23.0000	22.0000
78.2000	5.0000	23.0000	29.0000	28.0000
78.1333	5.0000	11.0000	23.0000	10.0000
78.1333	5.0000	10.0000	23.0000	25.0000
78.0667	5.0000	7.0000	23.0000	16.0000
78.0000	5.0000	7.0000	23.0000	20.0000
77.8667	5.0000	10.0000	23.0000	29.0000

Table [6.4] Results of combinations of 4 features

k	Correct classification	Performance Index
1	73	0.5196
2	74	0.5099
3	77	0.4796
4	77	0.4796
5	82	0.42
6	81	0.4359
7	76	0.4899
8	80	0.4472
9	79	0.4583
10	79	0.4583

Table[7.1] Classification results with changing K for the crisp classifier for set 1

k	Correct classification	Performance Index
1	74	0.5099
2	74	0.5099
3	77	0.4796
4	77	0.4796
5	74	0.5099
6	76	0.4899
7	76	0.4899
8	75	0.5000
9	78	0.4690
10	78	0.4690

Table[7.2] Classification results with changing K for the crisp classifier for set 2

k	Correct classification	Performance Index
1	79	0.4583
2	79	0.4583
3	81	0.4359
4	84	0.4000
5	83	0.4123
6	85	0.3873
7	81	0.4359
8	81	0.4359
9	82	0.4243
10	82	0.4243

Table[7.3] Classification results with changing K for the crisp classifier for set 3

k	Correct classification	Performance Index
1	75.3333	0.4959
2	75.6667	0.4927
3	78.3333	0.4650
4	79.3333	0.4531
5	79.6667	0.4474
6	80.6667	0.4377
7	77.6667	0.4719
8	78.6667	0.4610
9	79.6667	0.4505
10	79.6667	0.4505

Table[7.4] Average classification results with changing K for the crisp classifier

	percent classification						performanc e index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	73	73	73	73	73	73	0.5196
2	77	75	73	74	72	73	0.4267
3	75	74	77	75	73	69	0.4261
4	75	74	76	77	76	69	0.4157
5	74	74	81	79	76	73	0.4061
6	69	74	78	79	76	74	0.3993
7	70	74	77	81	77	72	0.3980
8	70	75	79	79	79	72	0.3977
9	69	72	78	80	79	71	0.3971
10	68	73	78	79	79	70	0.3978

Table[8.1] Classification results for the fuzzy classifier for set 1

	percent classification						performance index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	74	74	74	74	74	74	0.5099
2	72	75	74	77	78	77	0.4328
3	73	75	79	79	77	73	0.4316
4	73	75	79	76	76	72	0.4262
5	71	76	76	78	77	74	0.4176
6	72	73	76	79	75	72	0.4164
7	71	73	79	79	77	70	0.4092
8	69	74	78	80	77	70	0.4099
9	73	75	80	79	77	70	0.4059
10	72	73	81	79	76	72	0.4004

Table[8.2] Classification results for the fuzzy classifier for set 2

	percent classification						performance index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	79	79	79	79	79	79	0.4583
2	73	76	79	84	84	84	0.3991
3	72	75	81	85	85	82	0.3862
4	75	78	84	86	86	83	0.3704
5	74	80	83	86	86	84	0.3635
6	75	82	85	87	85	83	0.3588
7	74	80	82	84	84	82	0.3605
8	73	78	83	84	84	81	0.3638
9	73	79	83	84	85	81	0.3625
10	73	80	83	84	85	82	0.3615

Table[8.3] Classification results for the fuzzy classifier for set 3

	percent classification						performanc e index
k \ Threshold	0.3	0.4	0.5	0.6	0.7	0.8	
1	75.33	75.33	75.33	75.33	75.33	75.33	0.4959
2	74	75.33	75.33	78.33	78	78	0.4195
3	73.33	74.67	79	79.67	78.33	74.67	0.4146
4	74.33	75.67	79.67	79.67	79.33	74.67	0.4041
5	73	76.67	80	81	79.67	77	0.3957
6	72	76.33	79.67	81.67	78.67	76.33	0.3915
7	71.67	75.67	79.33	81.33	79.33	74.67	0.3892
8	70.67	75.67	80	81	80	74.33	0.3905
9	71.67	75.33	80.33	81	80.33	74	0.3885
10	71	75.33	80.67	80.67	80	74.67	0.3866

Table[8.3] Average classification results with for the fuzzy classifier

File	Membership	Defuzzified	Result
1.0000	0.2736	0	
2.0000	0.3339	0	
3.0000	0.5397	0	0
4.0000	0.5450	0	
5.0000	0.7423	1.0000	
6.0000	0.1732	0	0
7.0000	0.8901	1.0000	
8.0000	1.0000	1.0000	1 Misclassified
9.0000	0.5376	0	
10.0000	0.1742	0	
11.0000	0.4366	0	0
12.0000	0.3458	0	
13.0000	0.5145	0	
14.0000	0.5178	0	0
15.0000	0.1016	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.1334	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2923	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1607	0	0
25.0000	0	0	
26.0000	0.4421	0	
27.0000	1.0000	1.0000	0
28.0000	0.3307	0	
29.0000	0.0583	0	
30.0000	0.4965	0	0
31.0000	0.3505	0	
32.0000	0.1181	0	
33.0000	0.2101	0	0

Table [9.1] Classification of the files of set 1

File	Membership	Defuzzified	Result
34.0000	0.5970	0	
35.0000	0	0	
36.0000	0.1193	0	0
37.0000	0.3174	0	
38.0000	0.8117	1.0000	
39.0000	0.0997	0	0
40.0000	0.1889	0	
41.0000	0.4215	0	
42.0000	0.1635	0	0
43.0000	0.6474	1.0000	
44.0000	0	0	
45.0000	0.5495	0	0
46.0000	0.1115	0	0
47.0000	0	0	
48.0000	0.3986	0	
49.0000	0	0	
50.0000	0	0	0
51.0000	0.6709	1.0000	
52.0000	1.0000	1.0000	
53.0000	0.5297	0	1
54.0000	0.7245	1.0000	
55.0000	0.9200	1.0000	
56.0000	1.0000	1.0000	1
57.0000	0.9105	1.0000	
58.0000	0.9398	1.0000	
59.0000	0.5657	0	1
60.0000	0.8968	1.0000	
61.0000	1.0000	1.0000	
62.0000	0.2793	0	
63.0000	0.1088	0	0 **Misclassified**
64.0000	0.6245	1.0000	
65.0000	0.8643	1.0000	
66.0000	0.5054	0	1

Table [9.1] Continued

File	Membership	Defuzzified	Result
67.0000	0.8498	1.0000	
68.0000	0.6969	1.0000	
69.0000	0.8397	1.0000	1
70.0000	0.2901	0	
71.0000	0.8291	1.0000	
72.0000	0.3982	0	0 Misclassified
73.0000	1.0000	1.0000	
74.0000	0.2463	0	
75.0000	0.8043	1.0000	1
76.0000	0.6676	1.0000	
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	1
79.0000	1.0000	1.0000	
80.0000	0.7538	1.0000	
81.0000	1.0000	1.0000	1
82.0000	1.0000	1.0000	
83.0000	0.8378	1.0000	
84.0000	1.0000	1.0000	1
85.0000	0.8926	1.0000	
86.0000	0.5448	0	
87.0000	0.5751	0	0 Misclassified
88.0000	0.8273	1.0000	
89.0000	0.2945	0	
90.0000	0.9110	1.0000	1
91.0000	1.0000	1.0000	
92.0000	1.0000	1.0000	
93.0000	0	0	1
94.0000	0.2887	0	
95.0000	0.2079	0	
96.0000	0.5793	0	0 Misclassified
97.0000	1.0000	1.0000	
98.0000	0.7971	1.0000	
99.0000	0.8708	1.0000	1
100.0000	1.0000	1.0000	1

Table [9.1] Continued

File	Membership	Defuzzified	Result
1.0000	0.2579	0	
2.0000	0.1307	0	
3.0000	0	0	0
4.0000	0.2652	0	
5.0000	0.4345	0	
6.0000	0.1175	0	0
7.0000	1.0000	1.0000	
8.0000	0.7086	1.0000	1 Misclassified
9.0000	0.2856	0	
10.0000	0.2745	0	
11.0000	0.3056	0	0
12.0000	0.2720	0	
13.0000	0.5019	0	
14.0000	0.8871	1.0000	0
15.0000	0.0912	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.8334	1.0000	1 Misclassified
19.0000	0	0	
20.0000	0	0	
21.0000	0.5483	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.1535	0	0
25.0000	0.4955	0	
26.0000	0.1013	0	
27.0000	1.0000	1.0000	0
28.0000	0.3788	0	
29.0000	0.1638	0	
30.0000	0.0905	0	0
31.0000	0	0	
32.0000	0.1431	0	
33.0000	0.0937	0	0

Table [9.2] Classification of the files of set 2

File	Membership	Defuzzified	Result
34.0000	0	0	
35.0000	0	0	
36.0000	0.1281	0	0
37.0000	0.3690	0	
38.0000	0.5734	0	
39.0000	0.1569	0	0
40.0000	0.3659	0	
41.0000	0.4124	0	
42.0000	0.1704	0	0
43.0000	0.4251	0	
44.0000	0.0664	0	
45.0000	0.5356	0	0
46.0000	0.5084	0	0
47.0000	0.1735	0	
48.0000	0.7512	1.0000	
49.0000	0.5115	0	
50.0000	0.0976	0	0
51.0000	0.6361	1.0000	
52.0000	0.8482	1.0000	1
53.0000	0.3471	0	
54.0000	0.8822	1.0000	
55.0000	1.0000	1.0000	1
56.0000	1.0000	1.0000	
57.0000	1.0000	1.0000	
58.0000	0.8730	1.0000	1
59.0000	0	0	
60.0000	0.0389	0	
61.0000	0.3643	0	0 Misclassified
62.0000	1.0000	1.0000	
63.0000	0.8174	1.0000	
64.0000	0.8875	1.0000	1
65.0000	0.7995	1.0000	
66.0000	0.5919	0	
67.0000	0.7533	1.0000	1

Table [9.2] Continued

File	Membership	Defuzzified	Result
68.0000	0.7337	1.0000	
69.0000	0.8524	1.0000	
70.0000	0.8602	1.0000	1
71.0000	0.2217	0	
72.0000	1.0000	1.0000	
73.0000	0.1268	0	0 Misclassified
74.0000	0.8860	1.0000	
75.0000	0.2121	0	
76.0000	0.1684	0	
77.0000	0.6903	1.0000	0 Misclassified
78.0000	0.7680	1.0000	
79.0000	0.8735	1.0000	
80.0000	0.8013	1.0000	1
81.0000	0.1748	0	
82.0000	0.5428	0	
83.0000	0.8496	1.0000	0 Misclassified
84.0000	0.3444	0	
85.0000	0.8298	1.0000	
86.0000	0.8590	1.0000	1
87.0000	0.6879	1.0000	
88.0000	0.9082	1.0000	
89.0000	0.6653	1.0000	1
90.0000	0.1636	0	
91.0000	0.8754	1.0000	
92.0000	0.8594	1.0000	1
93.0000	0.5185	0	
94.0000	0.4932	0	
95.0000	0.7802	1.0000	0 Misclassified
96.0000	0.8684	1.0000	
97.0000	0.8788	1.0000	
98.0000	1.0000	1.0000	1
99.0000	1.0000	1.0000	
100.0000	0.8669	1.0000	1

Table [9.2] Continued

File	Membership	Defuzzified	Result
1.0000	0.3986	0	
2.0000	0.2845	0	
3.0000	0.2562	0	0
4.0000	0.2786	0	
5.0000	0.3226	0	
6.0000	0	0	0
7.0000	1.0000	1.0000	
8.0000	0.5055	0	
9.0000	0.1434	0	0
10.0000	0	0	
11.0000	0	0	0
12.0000	0.0691	0	
13.0000	0.4744	0	
14.0000	0.4708	0	0
15.0000	0	0	
16.0000	0	0	
17.0000	0	0	0
18.0000	0.4623	0	0
19.0000	0	0	
20.0000	0	0	
21.0000	0.2096	0	0
22.0000	0	0	
23.0000	0	0	
24.0000	0.0516	0	0
25.0000	0.2885	0	
26.0000	0.0981	0	
27.0000	0.9336	1.0000	0
28.0000	0.2254	0	
29.0000	0.1465	0	
30.0000	0.0680	0	0
31.0000	0	0	
32.0000	0	0	
33.0000	0.0939	0	0

Table [9.3] Classification of the files of set 3

File	Membership	Defuzzified	Result
34.0000	0.3917	0	
35.0000	0	0	
36.0000	0	0	0
37.0000	0.1689	0	
38.0000	0.5220	0	
39.0000	0	0	0
40.0000	0.0969	0	
41.0000	0	0	
42.0000	0	0	0
43.0000	0.4810	0	
44.0000	0.3154	0	
45.0000	0.4552	0	0
46.0000	0.3285	0	0
47.0000	0.3690	0	
48.0000	0.5593	0	
49.0000	0.3522	0	
50.0000	0.2325	0	0
51.0000	1.0000	1.0000	
52.0000	0.9052	1.0000	
53.0000	0.8115	1.0000	1
54.0000	0.8397	1.0000	
55.0000	0.8754	1.0000	
56.0000	0.0930	0	1
57.0000	0.8330	1.0000	
58.0000	1.0000	1.0000	1
59.0000	1.0000	1.0000	
60.0000	1.0000	1.0000	
61.0000	1.0000	1.0000	1
62.0000	1.0000	1.0000	
63.0000	0.6496	1.0000	
64.0000	0.5075	0	1
65.0000	0.0823	0	
66.0000	0.7810	1.0000	
67.0000	0.2356	0	0 Misclassified

Table [9.3] Continued

File	Membership	Defuzzified	Result
68.0000	1.0000	1.0000	
69.0000	1.0000	1.0000	
70.0000	1.0000	1.0000	1
71.0000	1.0000	1.0000	
72.0000	1.0000	1.0000	
73.0000	1.0000	1.0000	1
74.0000	1.0000	1.0000	
75.0000	1.0000	1.0000	
76.0000	1.0000	1.0000	1
77.0000	1.0000	1.0000	
78.0000	1.0000	1.0000	
79.0000	1.0000	1.0000	1
80.0000	0.6068	1.0000	
81.0000	0.9054	1.0000	
82.0000	0.4134	0	1
83.0000	1.0000	1.0000	
84.0000	0	0	
85.0000	0.2914	0	0 Misclassified
86.0000	1.0000	1.0000	
87.0000	1.0000	1.0000	
88.0000	0.8786	1.0000	1
89.0000	0.9018	1.0000	
90.0000	1.0000	1.0000	
91.0000	1.0000	1.0000	1
92.0000	1.0000	1.0000	
93.0000	0.9135	1.0000	
94.0000	0.8292	1.0000	1
95.0000	0.7423	1.0000	
96.0000	1.0000	1.0000	
97.0000	0.0902	0	1
98.0000	0.2564	0	
99.0000	0	0	
100.0000	0.4387	0	0 Misclassified

Table [9.3] Continued

Non deceptive	Deceptive 1	Deceptive 2	Deceptive 3
QQ8R9OIO.011	QQ4Q1O83.011	QQ7LX5Q0.021	QQ8RAJ0C.011
QQ8R9OIO.021	QQ4Q1O83.021	QQ7LX5Q0.031	QQ8RAJ0C.021
QQ8R9OIO.031	QQ4Q1O83.031	QQ7MN2Y0.011	QQ8RAJ0C.031
QQ95LU1T.011	QQ4Q3MDC.011	QQ7MN2Y0.021	QQ9EUKVT.011
QQ95LU1T.021	QQ4Q3MDC.021	QQ7MN2Y0.031	QQ9EUKVT.021
QQ95LU1T.031	QQ4Q3MDC.031	QQ7TC5UF.011	QQ9EUKVT.031
QQAURNUS.021	QQ51DE36.011	QQ7TC5UF.021	QQ9IOOXO.021
QQAURNUS.031	QQ51DE36.021	QQ7TC5UF.031	QQ9IOOXO.041
QQAV53P6.011	QQ51DE36.041	QQ7TQVER.011	QQ9SOW8L.011
QQAV53P6.021	QQ6RQGH6.011	QQ7TQVER.021	QQ9SOW8L.021
QQAV53P6.031	QQ6RQGH6.021	QQ7TQVER.031	QQ9SOW8L.031
QQBQ4SHI.011	QQ6RQGH6.031	QQ7TVADC.011	QQ9SQIK9.011
QQBQ4SHI.021	QQ6RQGH6.041	QQ7TVADC.021	QQ9SQIK9.021
QQBQ4SHI.031	QQ6T711O.011	QQ7TVADC.031	QQ9SQIK9.031
QQBSS7WT.011	QQ6T711O.021	QQ7U2T4R.011	QQ9W0B9F.011
QQBSS7WT.021	QQ6T711O.031	QQ7U2T4R.021	QQ9W0B9F.031
QQBSS7WT.031	QQ6Z59IG.011	QQ7U2T4R.031	QQ9W0B9F.041
QQ7OXM60.021	QQ6Z59IG.021	QQ7YP7QU.011	QQ9U4FMU.011
QQ7RH0RO.011	QQ6Z59IG.031	QQ7YP7QU.021	QQ9U4FMU.021
QQ7RH0RO.021	QQ7PP9B9.011	QQ7YP7QU.031	QQ9U4FMU.031
QQ7RH0RO.031	QQ7PP9B9.021	QQ7YZOJ3.011	QQ9Y_SVF.011
QQ7R51P9.011	QQ7PP9B9.031	QQ7YZOJ3.021	QQ9Y_SVF.021
QQ7R51P9.021	QQ7PDU1X.011	QQ7YZOJ3.031	QQ9Y_SVF.031
QQ7R51P9.031	QQ7PDU1X.021	QQ8_0DPT.011	QQ9YH3QF.011
QQ9TDSP3.011	QQ7PDU1X.031	QQ8_0DPT.021	QQ9YH3QF.021
QQ9TDSP3.021	QQ7_PIPF.011	QQ8_0DPT.031	QQ9YH3QF.031
QQ9TDSP3.031	QQ7_PIPF.021	QQ8_0DPT.041	QQA2TT4C.011
QQA8OWOI.011	QQ7_PIPF.031	QQ8_2UQ9.011	QQA2TT4C.021
QQA8OWOI.021	QQ7_JT70.011	QQ8_2UQ9.021	QQA2TT4C.031
QQA8OWOI.031	QQ7_JT70.021	QQ8_2UQ9.031	QQA3HIRX.011
QQBT22O6.011	QQ7_JT70.031	QQ800IG6.011	QQA3HIRX.021
QQBT22O6.021	QQ738DYX.011	QQ800IG6.021	QQA3HIRX.031
QQBT22O6.031	QQ738DYX.021	QQ800IG6.031	QQA32UTF.011
QQBO9O_9.011	QQ738DYX.031	QQ82OIU9.011	QQA32UTF.021
QQBO9O_9.021	QQ75ULP9.011	QQ82OIU9.021	QQA32UTF.031
QQBO9O_9.031	QQ75ULP9.021	QQ82OIU9.031	QQA6U_IF.011
QQBC7PP6.011	QQ75ULP9.031	QQ82SUTX.011	QQA6U_IF.031
QQBC7PP6.021	QQ79_EYF.011	QQ82SUTX.021	QQA6U_IF.041
QQBC7PP6.031	QQ79_EYF.021	QQ82SUTX.031	QQAM4E3L.011
QQCHCK_O.011	QQ79_EYF.031	QQ860ZNU.011	QQAM4E3L.021
QQCHCK_O.021	QQ7BGDML.011	QQ860ZNU.021	QQAM4E3L.031
QQCHCK_O.031	QQ7BGDML.021	QQ860ZNU.031	QQARF2_X.011
QQCDTKP0.011	QQ7BGDML.031	QQ89U_ZR.011	QQARF2_X.021
QQCDTKP0.031	QQ7ETC8I.011	QQ89U_ZR.021	QQARF2_X.031
QQCDTKP0.041	QQ7ETC8I.021	QQ89U_ZR.031	QQAWA38X.011
QQCM5Y56.011	QQ7ETC8I.031	QQ8ATU26.011	QQAWA38X.021
QQCQQT8Y.011	QQ7JAQCS.011	QQ8ATU26.021	QQAWA38X.031
QQCQQT8Y.021	QQ7JAQCS.021	QQ8ATU26.031	QQAYXZGU.011
QQCQQT8Y.031	QQ7JAQCS.031	QQ8FGMVI.011	QQAYXZGU.021
QQCQQT8Y.041	QQ7LX5Q0.011	QQ8FGMVI.021	QQAYXZGU.031

Table [10] NSA Polygraph files used in sets 1-3.

Note: Each set consists of non-deceptive files and one of the deceptive sets

This page left blank.

Appendix B:
Program Listings

Classify Program

```
% This is a Matlab program
% This script parses a matrix of polygraph
% vectors into training and testing vectors.
% It then calls the classifier, trains, tests
% and gives results.


c = 2;                    % number of classes
percent_train=.75;        % percentage of inputs used for training

features=[1]              % features to use
classification=1; % use fuzzy classifier
kk=5;                     % K in K nearest neighbor
change=1;                 % Randomize training and testing inputs
repeat=20;                % Number of repetitions
ut=.5;                    % Upper threshhold for 3 class fuzzy classifier
lt=.5;                    % Lower threshhold for 3 class fuzzy classifier

load set31;                 % file containing feature matrix
                                % and vector that indicates whether
                                   % column is truthful or deceptive
%classvect;                        % vector of classes eg. 1 = deceptive
                     % 0 = truthful vector
featurematrix = featmat;        % matrix of features
dimension = size(featurematrix);
columns = dimension(2);              % the total number of columns in the feature matrix
number_train = round(percent_train*columns);        % number of vectors
                                     % used for training


ur=.5;                        %upper threshold
continue=1;                          % to repeat the program
while (continue==1)
'=========================================================================='
apercent_classified=[];              % clear average results
acorrect=[];
acc=[];
ffresult=[];
ccresult=[];
ttestclass=[];

men=0;
while(men ~=7)
        men=menu('Select:','Features','Type','K','Random'...
        ,'Repeat','% training','Start','Defuzz','Exit');

        if (men==1)
        'enter a vector of the features you want tested (eg. [1 2 4]) '
```

```
        features = input(' ');            % features being tested
        end

        if (men==2)
                classification=menu('Type:','Fuzzy','Crisp');
        end

        if (men==3)
        kk = input('enter the "K" in K nearest neighbor   ')
        end

        if (men==4)
                change=menu('Selection','Random','Constant');
        end

        if (men==5)
                repeat=input('Enter number of repeatitions')
        end

        if (men==6)
                percent_train=input('Enter percentage of the files used for training, 1 for all-1')
        end
        if (men==8)
                ch=menu('Defuzzification', '3class', 'Upper thresh','Lower thresh');
                if ch==1,           classification=3,  end
                if ch==2
                        ut=input('enter the upper threshhold'); % lower limit for class 1
                end
                if ch==3
                        lt=input('enter the lower threshhold'); %upper limit for class 0
                end
        end
        if (men==9) break,end
end
if men==9 break,end
number_train = round(percent_train*columns);
acorrect=[];                % vector for the average of correct classification
acc=[];                     % vector for the average of performance index

if percent_train == 1       % To repeat nonrandom testing for all the files.
        repeat =columns;
end

for trial=1:repeat

featurematrix = featmat(features,:);    % creates a feature matrix of the
                                        % the features being tested
if ( (change==1) & (percent_train~=1) )
        [trainvect, testvect] = randvect(number_train,columns);
end;
if percent_train == 1
        testvect = trial;
        if (trial ==1)
                trainvect=2:columns;
```

```
        end
        if (trial == columns)
                trainvect=1:columns-1;
        end
        if ( trial ~=1  &  trial ~=columns )
                trainvect = [1:trial-1 , trial+1:columns];
        end
end
testvect
trainvect
u = featurematrix(:,testvect);          % testing matrix

testclass = classvect(1,testvect);       % class of each column in testing matrix

p = featurematrix(:,trainvect);          % training matrix

t = classvect(1,trainvect);             % class of each column in training matrix


if classification == 1                  % Fuzzy classifier

        % m = input('enter the degree of fuzziness "M" (1<=M<=infinfity)')
        m = 2;
        save fdatafil c kk m p t u
%       !fknn                           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')        %to make sure that the program actulally works
        dos('fknn|')
        'Now loading the result of the fuzzy classifier'
        load foutfile
        '-------------------------------------------------------'
        kk, features
        fresult
        testclass

        if(percent_train==1)
                ffresult=[ffresult fresult]
                ttestclass=[ttestclass testclass];
        end

        cr =fresult(2,:) > ut           % defuzzification of the result
        correct = 100*(1-mean(abs(testclass-cr)))  % percentage correct classified
        cc = [1-testclass; testclass];          % adding a row of complements to c
        cc=fresult-cc;
        'Performance Index='
        cc = sqrt(mean(mean(cc .^ 2)))
end

if classification == 2                  % crisp classifier

        save cdatafil c kk p t u
%       !cknn           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')                %to make sure that the program actulally works
        dos('cknn|')
        'Loading the Crisp output file'
```

```
        load coutfile
        '-----------------------------------------------------------'
        kk, features
        cresult
        testclass

        if(percent_train==1)
                ccresult=[ccresult cresult]
                ttestclass=[ttestclass testclass];
        end

        correct = 100*(1-mean(abs(testclass-cresult)))  % percentage correct classified
        cc = sqrt(mean(abs(testclass-cresult)))         % performance index

end
if classification == 3                  % Fuzzy classifier but defuzzification into 3 classes

        % m = input('enter the degree of fuzziness "M" (1<=M<=infinfity)')
        m = 2;
        save fdatafil c kk m p t u
%       !fknn                           %This line invokes the classifier program in a dos window
        dos('del foutfile.mat|')        %to make sure that the program actulally works
        dos('fknn|')
        'Now loading the result of the fuzzy classifier'
        load foutfile
        '-----------------------------------------------------------'
        kk, features
        fresult
        testclass

        if(percent_train==1)
                ffresult=[ffresult fresult]
                ttestclass=[ttestclass testclass];
        end
        class1=find(fresult(2,:) >ut);
        class0=find(fresult(2,:) <lt);
        class3=find(fresult(2,:) >lt & fresult(2,:) <ut);
        percent_classified=100*((length(class0)+length(class1))/length(testclass))
        fr=[fresult(:,class1) fresult(:,class0)]        % the section that is classified into one of the two
classes
        cr=fr(2,:)>ut
        tr=[testclass(class1) testclass(class0)]        % the section that is classified into one of the two
classes
        correct = 100*(1-mean(abs(tr-cr)))  % percentage correct classified
        cc = [1-tr; tr];        % adding a row of complements to cc
        cc=fr-cc;
        'Performance Index='
        cc = sqrt(mean(mean(cc .^ 2)))
end

        apercent_classified = [apercent_classified percent_classified]
        acorrect=[acorrect correct]
        acc=[acc cc]
```

```
end                  % for trial

if classification ==3                          % 3 class fuzzy
apercent_classified=mean(apercent_classified)
end
acorrect, mean(acorrect)
acc, mean(acc)

continue=3;
while (continue == 3 | continue==4)
continue=menu('Repeat?', 'Yes', 'no','Plot', 'threshold');
if(continue==3)
        dim=menu('Dimension', 'Two', 'Three')+1;
        if(dim==2)

                pp=p(:,find(t));
                plot(pp(1,:),pp(2,:),'r+');
                title('A clustering of two class data');
                hold on
                pp=p(:,find(t==0));
                plot(pp(1,:), pp(2,:), 'gx');

                pp=u(:, find(testclass));
                plot(pp(1,:), pp(2,:), 'r+');
                pp=u(:,find(testclass==0));
                plot(pp(1,:), pp(2,:), 'gx');

                hold off
        end     %if(dim==2)

        if(dim==3)

                pp=p(:,find(t));
                plot3(pp(1,:),pp(2,:), pp(3,:), 'r+');
                title('A clustering of two class data');
                hold on
                pp=p(:,find(t==0));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'rx');

                pp=u(:, find(testclass));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'g+');
                pp=u(:,find(testclass==0));
                plot3(pp(1,:), pp(2,:), pp(3,:), 'gx');

                hold off
        end     %if(dim==3)

  end    %if(continue==3)

  if (continue==4)

        ch=menu('Defuzzification', '3class', 'Upper thresh','Lower thresh');
        if ch==1,          classification=3,  end
```

```
        if ch==2
                ut=input('enter the upper threshhold'); % lower limit for class 1
        end
        if ch==3
                lt=input('enter the lower threshhold'); %upper limit for class 0
        end

        if classification==1
                cr =ffresult(2,:) > ut          % defuzzification of the result
                correct = 100*(1-mean(abs(ttestclass-cr)))  % percentage correct classified
                cc = [1-ttestclass; ttestclass];          % adding a row of complements to c
                cc=ffresult-cc;
                'Performance Index='
                cc = sqrt(mean(mean(cc .^ 2)))
        end

        if classification==2
                correct = 100*(1-mean(abs(ttestclass-ccresult)))  % percentage correct classified
                cc = sqrt(mean(abs(ttestclass-ccresult)))        % performance index
        end

        if classification==3
                class1=find(ffresult(2,:) >ut);
                class0=find(ffresult(2,:) <lt);
                class3=find(ffresult(2,:) >lt & ffresult(2,:) <ut);
                fr=[ffresult(:,class1) ffresult(:,class0)]        % the section that is classified into one of
the two classes
                cr=fr(2,:)>ut
                tr=[ttestclass(class1) ttestclass(class0)]        % the section that is classified into one of
the two classes
                percent_classified=100*((length(class0)+length(class1))/length(ttestclass))
                correct = 100*(1-mean(abs(tr-cr)))  % percentage correct classified
                cc = [1-tr; tr];          % adding a row of complements to cc
                cc=fr-cc;
                'Performance Index='
                cc = sqrt(mean(mean(cc .^ 2)))
        end
end

end     % while continue == 3 | 4

end                     % while continue
```

```
/*      This program implements a K-nearest neighbor classifier.
        created by:  Shahab Layeghi

        created:  8/4/93
        last modified: 9/17/93

*/
```

```
/* The main program opens a matlab data file, reads the training matrix,
classifies each entry in the testing matrix, and writes the result in an
output file. The file that this program gets the information from should be
called  "cdatafil.mat". As the name implies it is in matlab file format.
The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A trainig matrix 'P' which contains a set of feature vectors. Each vector
is in a column of the matrix.
4. A classes vector 'T' which contains the classes of the training set
5. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the CrispKNN routine to classify each one of the input
vectors and saves the results (the classes that these inputs belong to) in a
file called coutfile.mat.  This file is in Matlab format.  This file contains
a vector of the classes called:

'cresult'


This program can be called from dos, or within Matlab by using dos escpae
character '!'.  An example Matlab script file that shows how this program can
be used is included in the file "cknntest.m".

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <conio.h>

#define INPUTFILE "cdatafil.mat"
#define OUTPUTFILE "coutfile.mat"

// Function Prototypes ---------------------------------------------

int CrispKNN(double *Input, double *Samples, double *Lables);
double FindDistance(double *vec1, double *vec2);
double Maxd(double *vec, int *index, int Length);
int FindMax(int *vector, int *count, int Length, int Max);
int loadmat(FILE * fp,int *type, char *pname, int *mrows, int *ncols,
                    int *imagf, double **preal, double **pimag);
void savemat(FILE *fp, int type, char *pname, int mrows, int ncols,
                    int imagf, double *preal, double *pimag);


// Global variables, these variables will be set by reading matlab file  ------

int classes;          /* the number of classes */
int features;         /* Number of features in a class */
```

```
int KK ;                                    /* K in K-nearest neighbors */
int SampleSize;                             /* Number of Labled Samples */
int TestSize;

//-------------------------------------------------------------------


/*------------------------------------------------------------------*/

void main()
{

        double *Lables;
        double *KP;
        double *input;
        int i,j;
        FILE *fp;
        char name[20];
        int type, imagf;
        double *Samples, *isamples;    // isamples is for imaginary part of the matrix that is not used in
here.
        double *Testdata;
        double *result;
        fp=fopen(INPUTFILE,"rb");
        if(!fp) {
                printf("cannot open the file");
                exit(-1);
        }
        // read classes from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include classes at the beginning of the file\n");
                exit(-1);
        }
        classes=*KP;

        // read KK from the file
        loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
        if(i!=1 || j!=1) {
                printf("error: You should include K at the beginning of the file\n");
                exit(-1);
        }
        KK=*KP;

        // read the matrix from the datafile.
        loadmat(fp, &type, name, &features, &SampleSize, &imagf, &Samples, &isamples);

        // reading lables from data file
        loadmat(fp, &type, name, &i, &j, &imagf, &Lables, &isamples);
        if(i!=1 || j!=SampleSize) {
                printf("error: Number of labels is different from the number of samples\n");
                exit(-1);
        }
```

```
        // read data to be classified from the file
        loadmat(fp, &type, name, &i, &TestSize, &imagf, &Testdata, &isamples);
        if(i != features) {
                printf("error: Training and testing matrices should have the same size");
                exit(-1);
        }

        // Allocate space for result vector

        result = (double *) malloc(TestSize*sizeof(double));
        if(!result) {
                printf("Error: cannot allocate memory for the result vector");
                exit(-1);
        }

        for(i=0; i<TestSize; i++) {          // for each input
                input=Testdata+i*features;
                result[i]=CrispKNN(input, Samples, Lables);
//              printf("class: %lf\n", result[i]);
        }
        fclose(fp);
//      printf("\n  End of classification, Now writing the result in the file");

        fp=fopen(OUTPUTFILE, "wb");
        if(!fp) {
                printf("Error: Cannot write the file");
                getch();
        }
        savemat(fp, 0, "cresult", 1, TestSize, 0, result, result);
        fclose(fp);

}

/*--------------------------------------------------------------*/
int CrispKNN(double *Input, double *Samples, double *Lables)
{
        int i,j ;
        int nj, k, nk;
        double *distance;
        int *index;
        double x,y;

        distance = (double *) malloc(KK*sizeof(double));
        if(!distance) {
                printf("Error: Not enough memory for distance vector");
                exit(-1);
        }

        index = (int *) malloc(KK*sizeof(int));
        if(!index) {
                printf("Error: Not enough memory for index vector");
                exit(-1);
        }
```

```
for(i=0; i<KK; i++) {  // This loop initializes K nearest neighbors to the first K Samples
        index[i]=Lables[i]+1;
        distance[i]=FindDistance(Input, &Samples[i*features]);
}
for(i=KK; i<SampleSize; i++) {   // This is the loop that finds the K nearest Neighbors
        x=Maxd(distance, &j, KK);
        y=FindDistance(Input, &Samples[i*features]);
        if(y < x) {  // This sample is closest to the input than the farthest K Neighbors
                distance[j]=y;
                index[j]=Lables[i]+1;
        }
}
j=FindMax(index, &nj, KK, classes);  // Finds the class of maximum occurance

/* In this section it is checked to see if there is a tie.  That is if
there are two or more classes with the same number of occureances.  If
there is a tie for two classes,  the class with the minimum sum of
distances is selected. No action is taken for a tie of more than two
classes.  */

for (i=0; i<KK; i++)
        if(index[i]==j) index[i]=0;
k=FindMax(index, &nk, KK, classes);
if(nk==nj) {                    // If there is a tie.
        x=0;
        for(i=0; i<KK; i++) {
                if(index[i]==0)
                        x+=distance[i];
        }
        y=0;
        for(i=0; i<KK; i++) {
                if(index[i]==k)
                        y+=distance[i];
        }
        if(y<x)                                         //If sum of the distances to class j is
less than that of class k
                j=k;
}

free(distance);
free(index);
return j-1;

}

/*---------------------------------------------------------------------*/
/* This function returns the Euclidian distance between two vectors */

double FindDistance(double *vec1, double *vec2)
{

        int k;
        double distance;
```

```
        distance = 0;
        for(k=0; k<features; k++) {
                        distance +=(vec1[k]-vec2[k])*(vec1[k]-vec2[k]);
//                      distance += pow(vec1[k]-vec2[k] , 2);
        }
        return distance;
}

/*----------------------------------------------------------------*/
/* This function finds the biggest element of an array.  It returns that
value and also returns the index to that element in index.
*/

double Maxd(double *vec, int *index, int Length)
{
        int i,j=0;

        j=0;
        for(i=1; i<Length; i++)
                if(vec[i]>vec[j]) j=i;
        *index=j;
        return(vec[j]);
}

/*--------------------------------------------------------------------*/
/* This function finds a number that is most often repeated in an array of
integer values, and returns that number. Length of array shoud be less than
100.  It is supposed that number is an integer greater than zero.
vector is a pointer to the array.  count is the number of times that the
number is repeated.  Length is the length of the vector.
*/

int FindMax(int *vector, int *count, int Length, int Max)
{

        int i, j, m;
        int t[101];

        if(Max>100) Max=100;
        for(i=0; i<Max+1; i++)
                t[i]=0;
        for(i=0; i<Length; i++)
                t[vector[i]]++;
        m=t[1];
        j=1;
        for(i=1; i<Max+1; i++) {
                if(t[i]>m) {
                        m=t[i];
                        j=i;
                }
        }
        *count=m;
        return (j); }
```

```
/*      This program implements a fuzzy version of K-nearest neighbor classifier.
        created by:  Shahab Layeghi

        created:  9/1/93
        last modified: 9/3/93

*/

/* The main program opens a matlab data file, reads the training matrix,
classifies each entry in the testing matrix, and writes the result in an
output file. The file that this program gets the information from should be
called  "fdatafile.mat". As the name implies it is in matlab file format.
The data in this file should have the following order:

1. A single variable 'C' which is the number of classes.
2. A single variable 'K' which is the parameter 'K' in K-NN Algorithm.
3. A single variable 'M' which is the coefficient in fuzzy algorithm.
4. A trainig matrix 'P' which contains a set of feature vectors. Each vector
is in a column of the matrix.
5. A class membership matrix 'T' which contains the membership values of the
training set vectors to the classes.
6. An input matrix 'U' which contains a set of unclassified feature vectors.

The main program uses the FuzzyKNN routine to classify each one of the input
vectors and saves the results (the classes that these inputs belong to) in a
file called "foutfile.mat". This file is in Matlab format.  This file contains
a single variable called fresult.  It is a vector of the classes.

This program can be called from dos, or within Matlab by using dos escpae
character '!'.  An example Matlab script file that shows how this program can
be used is included in the file "fknntest.m".

*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
#include <conio.h>

#define INPUTFILE "fdatafil.mat"
#define OUTPUTFILE "foutfile.mat"

// Function Prototypes ---------------------------------------------

void FuzzyKNN(double *Input, double *Samples, double *Lables, double *Result);
double FindDistance(double *vec1, double *vec2);
double Maxd(double *vec, int *index, int Length);
int FindMax(int *vector, int *count, int Length, int Max);
int loadmat(FILE * fp,int *type, char *pname, int *mrows, int *ncols,
                    int *imagf, double **preal, double **pimag);
void savemat(FILE *fp, int type, char *pname, int mrows, int ncols,
```

```
// read M from the file
loadmat(fp, &type, name, &i, &j, &imagf, &KP, &isamples);
if(i!=1 || j!=1) {
        printf("error: You should include M as the thrid parameter\n");
        exit(-1);
}
M=*KP;

// read the matrix from the datafile.
loadmat(fp, &type, name, &features, &SampleSize, &imagf, &Samples, &isamples);

// reading lables from data file
loadmat(fp, &type, name, &i, &j, &imagf, &Lables, &isamples);
if(i!=1 || j!=SampleSize) {
        printf("error: Number of labels is different from the number of samples\n");
        exit(-1);
}

// read data to be classified from the file
loadmat(fp, &type, name, &i, &TestSize, &imagf, &Testdata, &isamples);
if(i != features) {
        printf("error: Training and testing matrices should have the same size");
        exit(-1);
}

// Allocate space for result vector

result = (double *) malloc(TestSize*Classes*sizeof(double));
if(!result) {
        printf("Error: cannot allocate memory for the result Matrix");
        exit(-1);
}

for(j=0; j<TestSize; j++) {          // for each input
        input=Testdata+j*features;
        FuzzyKNN(input, Samples, Lables, iresult);
//      printf("\n Memberships:");
        for(i=0; i<Classes; i++) {
                result[j*Classes+i]=iresult[i];
                printf(" %lf ", iresult[i]);
        }
}
fclose(fp);
//      printf("\n  End of classification, Now writing the result in the file");

fp=fopen(OUTPUTFILE, "wb");
if(!fp) {
        printf("Error: Cannot write the file");
        getch();
}
savemat(fp, 0, "fresult", Classes, TestSize, 0, result, result);
fclose(fp);
```

```
}

/*--------------------------------------------------------------------*/
/* This is a fuzzy K Nearest neighbor classifier routine.  Input is the
vector to be classified, Samples is the matrix of classified samples,
Lables is the vector of the classes that these samples belong to.
Result is the vector of membership values of Input to each class.
*/
void FuzzyKNN(double *Input, double *Samples, double *Lables, double *Result)
{
	int i,j,n ;
	int nj, k, nk;
	double *distance;
	int *index;
	double x,y;
	double *membership;		// pointer to membership matrix
	double nsum, dsum, temp;

	/* This section builds a fuzzy membership matrix from the lables.
	Membership of each sample to the class that it belongs to is assigned
	to 1, and the membership of it to other classes is assigned to 0 */

	membership = (double *) malloc(SampleSize*Classes*sizeof(double));
	if(!membership) {
		printf("Error: Not enough memory for membership matrix");
		exit(-1);
	}
	for(i=0; i<SampleSize*Classes; i++)
		*(membership+i)=0;		// Initializing matrix to zero
	for(j=0; j<SampleSize; j++) {
		i=*(Lables+j);
		*(membership+i*SampleSize+j)=1;
	}

	distance = (double *) malloc(KK*sizeof(double));   // allocate space for the vector
	if(!distance) {
		printf("Error: Not enough memory for distance vector");
		exit(-1);
	}

	index = (int *) malloc(KK*sizeof(int));
	if(!index) {
		printf("Error: Not enough memory for index vector");
		exit(-1);
	}

	for(i=0; i<KK; i++) {  // This loop initializes K nearest neighbors to the first K Samples
		index[i]=i;
		distance[i]=FindDistance(Input, &Samples[i*features]);
	}
	for(i=KK; i<SampleSize; i++) {    // This is the loop that finds the K nearest Neighbors
		x=Maxd(distance, &j, KK);
		y=FindDistance(Input, &Samples[i*features]);
		if(y < x) {  // This sample is closest to the input than the farthest K Neighbors
```

```
                        distance[j]=y;
                        index[j]=i;
                }
        }
        for(j=0; j<Classes; j++) {
                nsum=dsum=0;
                for(n=0; n<KK; n++) {
                        i=index[n];
                        temp=FindDistance(Input, &Samples[i*features]);
                        if(temp < 1e-10)  {                                         //If distance is
zero
                                Result[j]=membership[j*SampleSize+i];
                                break;
                        }
                        if(M == 2)
                                temp=1/temp;
                        else if(M != 1)
                                temp=pow(1/temp, 1/(M-1));
                        else
                                temp=0;
                        nsum += membership[j*SampleSize+i]*temp;
                        dsum += temp;
                }
                if(dsum !=0)
                        Result[j]=nsum / dsum;
        }
        free(membership);
        free(distance);
        free(index);
}


/*--------------------------------------------------------------------------*/
/* This function returns the Euclidian distance between two vectors */

double FindDistance(double *vec1, double *vec2)
{

        int k;
        double distance;

        distance = 0;
        for(k=0; k<features; k++) {
                        distance += (vec1[k]-vec2[k])*(vec1[k]-vec2[k]);
//                      distance += pow(vec1[k]-vec2[k] , 2);
        }
        return distance;
}

/*--------------------------------------------------------------------------*/
/* This function finds the biggest element of an array.  It returns that
value and also returns the index to that element in index.
*/
```

```
double Maxd(double *vec, int *index, int Length)
{
        int i,j=0;

        j=0;
        for(i=1; i<Length; i++)
                if(vec[i]>vec[j]) j=i;
        *index=j;
        return(vec[j]);
}

/*--------------------------------------------------------------------*/
/* This function finds a number that is most often repeated in an array of
integer values, and returns that number. Length of array shoud be less than
100.  It is supposed that number is an integer greater than zero.
vector is a pointer to the array.  count is the number of times that the
number is repeated.  Length is the length of the vector.
*/

int FindMax(int *vector, int *count, int Length, int Max)
{

        int i, j, m;
        int t[101];

        if(Max>100) Max=100;
        for(i=0; i<Max+1; i++)
                t[i]=0;
        for(i=0; i<Length; i++)
                t[vector[i]]++;
        m=t[1];
        j=1;
        for(i=1; i<Max+1; i++) {
                if(t[i]>m) {
                        m=t[i];
                        j=i;
                }
        }
        *count=m;
        return (j);
}
```

Report No. DoDPI96-R-0002

Use of Fuzzy Set Classification for Pattern Recognition of the Polygraph

Ramin Djamschidi
San Jose State University
Department of Electrical Engineering
San Jose, CA 95106

September 1994

Department of Defense Polygraph Institute
Fort Mcclellan, AL 36205

ACKNOWLEDGMENT

While doing this project, I have been fortunate to receive great assistance, suggestions and support from numerous students, colleagues and friends. It was a wonderful experience for me to work and live with each and every one of them. Not to mention, the huge amount of the encouraging e-mails.

I would like to thank Jürgen Niemeyer, Elmar Bongers and Robert Hilbing for their continuos and reliable help from Germany (you guys took away - more than once - some big burdens from my shoulder). How much I enjoyed Bob's e-mails. (Thank you for sharing with me your life-changing experiences...). I would like to thank Tim for helping me set the foundation of where I stand in my life.

I was surrounded by an awesome 'crowd' of people in my lab, beginning with Raghu Kondapalli who was always there with his practical and honest helps with an additional portion of encouraging humor. In these very last seconds of typing my report, (you are sitting in front of me and trying to fix a table for me - it is 3:00a.m!) I realize the privilege of getting you to know. I would also thank Shahab for his initial advice and for being there for me as a friend and also Mitra for her irreplaceable help for figuring out the names of the features. How ashamed I was to bother her so often.

I would also like to thank those who "officially" were set apart to help me: As they were Chuck Lam, Ulka Agarwal and Michelle Badal. How could I have ever accomplished my thesis in this quality without their assistance? I would specially like to express my thanks to Chuck for his significant role by the LMS part of my project. (All the approvals I might get for it is completely yours.)

I would like to express my gratitude to my former supervisor Prof. Duda who was always there for a "genius"-advice. Without his initial invitation for my first project, I could not have come to SJSU. I would also thank Melinda and Lois in the EE-office for their humorous and reliable helps. I would never forget the very first days when I came here as a stranger and met them, (How could they remember my name...?) I would also thank Phelomena - in the nicest building of SJSU where my grant came in - for her endless time to help me fill all the forms. I do not know exactly how many people were actually involved in the process of supporting me financially till the actual grant came.

I cannot finish this list without mentioning Dolat and Parviz for their patience and care throughout their most challenging phase of life. (By the way, how many months did I actually live at your home?)

Last but not least, I would like to express my highest gratitude to my supervisors in Aachen and San Jose who **actually** made all these possible for me. I really appreciate Prof. Rau's efforts to lead the necessary procedures for my project at SJSU. How much encouraging was that for me when he sent the very initial fax for my thesis at the second day of his vacation. I would also like to thank my direct supervisor Dr. Thull for his continuos supports and tips. He never forgot to add some sparkling statements to his emails (and those from Marian!). I hope I filled his expectations.

And finally, Prof. Knapp's wonderful companionship combined with the unlimited freedom of decision he gave me, is the major reason of this project's success. Not to mention, his efforts to help me to bridge the financial gap till I received my grant and his humorous comforts when something went wrong. It was an irreplaceable experience to work with him.

Table of Contents

List of Figures

§2. INTRODUCTION

2.1. POLYGRAPH[1]

2.1.1. Preview:

Polygraph examinations are the most widely used method to distinguish between truth and deception. In a Polygraph examination a person is connected to a special instrument called a Polygraph which records several physiological signals such as blood pressure, Galvanic Skin Response, and respiration. The subject is asked a set of questions by an examiner. By looking at these signals the examiner is able to determine the reactions of the subject to the questions and decide whether the person was truthful or deceptive in answering each question.

The problem with human classification of Polygraph tests is that the outcome depends on the examiner's experience and personal opinion. Automatic scoring of Polygraph tests has been a subject of extensive research. Several methods for Polygraph classification have been studied which are mostly based on *statistical* classification techniques.

Digitized Polygraph data used in this project were collected from various police stations. The data files were organized according to the test format used and were decoded to ASCII format so they can be read by Matlab. Preprocessing and feature extraction routines were implemented in the Matlab language in privious works [Layeghi1993,1] [Dastmalchi1993][Jacobs1993]. Three sets of files were chosen, each one of them contained 50 deceptive and 50 non-deceptive files.
These files are listed in the appendix, Fig.42.

2.1.2. History:

The first attempt to use a scientific instrument in an effort to detect deception occurred around 1895 [Reid1966]. That was the year that Caesar Lombroso published the results of his experiments in which a hydrosphygmograph was used to measure the blood pressure-pulse changes of criminals in order to determine whether or not they were deceptive. Although the hydrosphygmograph was originally intended to be used for medical

[1]Portions of this section were extracted from [Layeghi1993,1] using particularly [Capps1992] [Olsen1983] [Reid1966].

purposes, Lombroso found that it worked well for lie detection. Lombroso may have been the first to use a peak of tension test format. This was done by showing a suspect a series of photographs of children, one being the victim of sexual assault. If the suspect did not react more to the victims picture than the pictures of the other children, Lombroso concluded that the suspect did not know what the victim looked like and therefore was not the alleged perpetrator.

In 1914 Vittorio Benussi published his research on predicting deception by measuring recorded respiration tracings [Capps1992]. He found that if the length of inspiration were divide by the length of expiration, the ratio would be larger after lying than before lying and also before telling the truth than after telling the truth. In 1921 John A. Larson constructed an instrument capable of simultaneously recording blood pressure pulse and respiration during an examination [Reid1966] [Capps1992]. Larson reported accurate results which prompted Leonarde Keeler to construct a better version of this instrument in 1926 [Reid1966] [Capps1992].

The use of galvanic skin response in lie detection began during the turn of the century. It's usefulness, however, did not become evident until the 1930's during which time several articles written by Father Walter G. Summers of Fordham University in New York [Capps1992]. In these articles he reports over 90 criminal cases in which examination using the galvanic skin response had all been successful and confirmed by confession or supplementary evidence.

The usefulness of the galvanic skin response prompted Keeler to add an galvanometer to his polygraph. At the time of Keelers death in 1949, the Keeler Polygraph recorded blood pressure-pulse, respiration, and galvanic skin response [Reid1966].

2.1.3. Modern Test Formats:

The effectiveness of a polygraph examination is often the result of the test format that is used. A polygraph test format consists of an ordered combination of relevant questions about an issue, control questions that provide a physical response for comparison, and irrelevant questions that also provide a response or the lack of a response for comparison [Olsen1983][Capps1992].

Three general types of test formats are in use today. These are *Control Question Tests, Relevant-Irrelevant Tests,* and *Concealed Knowledge Tests.* Each of the general test formats may have a number of more specific variations. Each examination consists of two to five sessions containing a prescribed series of questions. The test format that is used in an examination is determined by the test objective [Reid1966] [Capps1992].

1. The *Concealed Knowledge Test,* also called peak of tension test, is used when facts about a crime are known only by the investigators and not by the public. In this case, a subject would not know the facts unless he or she was guilty of the crime. For example, if a gun was used in a crime and the public did not know the caliber, an examiner could ask a suspect, if it was a 22 caliber, a 38 caliber, or a 9 mm. If the gun used was a 9 mm and the suspect was deceptive, a polygraph chart would probably indicate evidence of deception.

2. A *Control Question Test*[2] is often used in criminal investigations. In this type of test a series of relevant, irrelevant, and control questions are asked:

- A relevant question is one which is specific to the crime being investigated. For example, "Did you steal the money?".

- A control question is designed to make the subject feel uncomfortable. It is not specific to the crime being investigated however it may be related in an indirect way. A control question that could follow the relevant question stated above is "Have you ever taken anything that did not belong to you?". The control questions are compared to the relevant questions and if the responses to the relevant questions are greater, the subject is usually classified as deceptive.

- Irrelevant questions are used as buffers. Examples of irrelevant questions are "Are the lights in this room on?" or "Is today Monday?".

3. *Relevant-Irrelevant Tests* are usually used to test people trying to obtain security clearance or get a job. In this test, relevant questions are compared to irrelevant questions. Very few control questions are asked. The purpose of control questions in this test is to make sure that the subject is capable of reacting at all.

---

[2] It was decided to use this method in our project (as it was also in previous works).

2.1.4. Present Day Equipment

The most popular polygraph machines today are the Reid Polygraph developed in 1945 and the Axciton Systems computerized polygraph developed in 1989 [Olsen1983]. The Reid polygraph scrolls a piece of paper under pens that record the biological signals. The Axciton polygraph digitizes physiological signals and uses a computer to process them. The sampling frequency of the Axciton machine is 30 Hz. Axciton provides a computer based system for ranking the subject responses but allows printouts of the charts to be scored by hand the traditional way.

Both machines record the same biological signals using standard methods. Blood pressure is measured by placing a standard blood pressure cuff on the arm over the brachial artery. Respiration is monitored by placing rubber tubes around the abdominal area and the chest of the subject. This results in two signals, a lower and upper respiratory signal. Skin conductivity is measured by placing electrodes on two fingers of the same hand.

The focus of this thesis is to investigate two different fuzzy pattern recognition algorithms using the aforementioned signals.

2.2. PATTERN RECOGNITION UTILIZING FUZZY TOOLS

2.2.1. Why the "FUZZY" approach?

While observing the history of science, we notice that one of its major goals has always been what we call today "pattern recognition". Having this in mind, man created models, functional relationships and mathematical tools to come closer to a perfect and precise model for almost every area of the nature and our being. In fact, *"precision"* became more and more important, to the extent that an *imprecise* model was a *bad* model by default.

1965 Lotfi A. Zadeh introduced in his innovative paper [Zadeh1965] an *"imprecise"* structure for mathematical observation; Hence, the *fuzzy set* was born. A companion to the classical one with often more useful and suitable representation of our environment.

"The fuzzy set was conceived as a result of an attempt to come to grips with the problem of pattern recognition in the context of imprecisely defined categories. In such cases, the belonging of an object to a class is a matter of degree, as is the question of whether or not a group of objects form a cluster"; These were the introductory words from L.A. Zadeh in [Bezdek1981]. They summarize the fundament of any fuzzy clustering or classifying algorithm concerning any search of data structure or pattern recognition. This concept is exactly what this project is all about.

An example:
Imagine, you have two groups of objects "chairs" and "desks" in different varieties. In a simple version of a typical pattern recognition problem, you have the task to cluster or classify the given objects into these two groups. In reality, we will also have other objects like a big box or a bed within the pool of the objects, but only the two aforementioned clusters by definition. Now, a conventional crisp clustering method would put these critical objects in either one of these two clusters. Thus, the big box or the bed may be labeled as if they would be chairs.
A fuzzy clustering method would label the objects with *soft* membership values. In this case, a big box (that can be used as a chair or a desk) might be labeled with 0.6 degree chair and 0.4 desk. Information like this serves a useful purpose - "fuzzy memberships in several classes are a signal to take a second look" [Bezdek1993] [Bezdek1992]:

Hard memberships of data cannot support this. Thus, the fuzzy model provides a richer and more flexible solution structure, one that models the real objects with a finer degree of detail than the harshness of the crisp models. Notice also that hard membership values build a subset of the fuzzy membership[3] set.

There are different types of fuzzy algorithms to find the appropriate membership values within the data. In this project, we used the follwoing two approaches:

1. Clustering algorithms:

Given any finite data, the problem of clustering is to find similarities between the objects of the data and to assign labels that matching objects would belong to the same subgroups. The algorithm starts its search without any initial interpretative information about the data elements. It *only* seeks for objective numerical similarities between the elements. Because the initial objects are unlabeled, this method is often called "unsupervised learning". The word *learning*[4] implies that the clustering algorithm will ultimately find the correct labels at the end of the process. This is what we hope to obtain, but we do not know it a priori.

Notice that because of the unsupervised nature of this algorithm, we may find "correct" clusters which represent some similarities, but not the ones we were looking for. In the aforementioned example with chairs and desks, the algorithm may provide two clusters of "wood-made" and "metal-made" objects (which are also correct), but not "chairs" and "desks" as we had hoped for.

In this case, the performance of a clustering model is influenced by the choice of the parameters[5], features, geometrical properties and our eventual interpretation of the labels.

2. Classifying algorithms:

In contrast to a clustering system which labels a given data, a classifier is capable - once it is defined (and trained) - of labelling every appropriate data. In addition, a classifying system is *ususally* initialized by labeled objects. In these cases, we call this method "supervised learning".

[3]Notice that membership values are *not* probabilities; they are similarities of object vectors to a class structure. They represent the degree of belonging of an object to a group of objects.

[4]The word *learning* does not imply any *training*. In fact, a clustering system - as is its nature - is almost the opposite of any system which *learns* by *training*.

[5]See chapter 2.2.3.2. for the meanings of the parameters and chapter 3.1.3.3. for the strategies we used.

Notice that we can also use a clustering algorithm as a modified classifying algorithm: After having set the optimal combination of parameters and features, we can use the clustering system to classify any new data by:

- adding the new element to a given and already correct clustered data, and letting the system *re*label[6] the data. Thus, our new object ends up to be in one of the clusters representing its identity,

- saving all the parameters, cluster centers and the data elements and calculate appropriately the membership value of the new object, which will eventually represent its identity.

[6]Running a new clustering process with one more element will probably change the structure of the original clusters, because the cluster centers and the membership values of each element depend on *all* of the members. In spite of this fact, we will be able to classify a normal (= not an outlier) object by having a large number of already clustered objects in a stable condition.

2.2.2. Why fuzzy-c-means (FCM)?

One of the most significant characteristic of *fuzzy*-c-means algotithm is its "fuzziness"[7], as the name assumes. Unlike crisp clustering methods, FCM gives us "membership functions" $\subset [0, 1]$ which determine the *grade* of belongingness of the elements to a cluster. As mentioned before, this information is totally lost by conventional clustering techniques. The advantage of FCM is the fact that the results we may get from a crisp clustering method are automatically within those from FCM.

We chose FCM as an alternative and a comparison to the fuzzy K-Nearest-Neighbor algorithm (KNN) investigated previously [Layeghi1993,1][Dastmalchi1993][Jacobs1993], specially because FCM is an *unsupervised* clustering method which works only by using "mathematical" tools such as spatial distances or similarities, without any training or additional interpretative information.
By this method, good[8] features will then hopefully provide an optimal mathematical grouping that presents in some sense an accurate portrayal of natural structures in the physical process from where the polygraph data are drived.

Why we chose FCM algorithm:

Because it

- **does not need previous training,**
- **does not make any assumption about the distribution of samples,**
- **is unsupervised, objective and self organized,**
- **can be used as an alternative and a comparison to fuzzy KNN investigated previously.**

Fig.1: FCM characteristics

[7]See chapter 2.1.1. for characteristics of a fuzzy approach.
[8]"Good" features are in our study those which can cluster the data in deceptive and truthful groups.

2.2.3. Fuzzy-c-means algorithm and its interpretation

2.2.3.1. FCM code - An iterative procedure:

The fuzzy-c-means algorithm[9] is basically an iterative procedure to minimize an objective function J_m representing a spatial fuzzy distance between data points x_k and cluster centers v_i. In this project, I chose the most widely used Euclidean distance, i.e. the sum of the squared errors performance index;

$$\boxed{J_m(U,v)=\sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m\left\|x_k-v_i\right\|_A^2}$$

- $X=\{x_1,x_2,\dots,x_n\}\subset\Re^s$ is a finite data set in the pattern space $\Re^s$.
- c is a fixed and known number of clusters (here: c=2).
- $U=[u_{ik}]\in\Re^{cn}$ is a fuzzy c-partition of X, u_{ik} is referred to as the grade of membership of x_k to the cluster i. u_{ik} satisfy the following constraints;

$$u_{ik}\in[0,1];1\le i\le c,1\le k\le n$$

$$\sum_{i=1}^{c}u_{ik}=1;1\le k\le n$$

$$0<\sum_{k=1}^{n}u_{ik}<n;1\le i\le c$$

- $V=(v_1,v_2,\dots,v_c)\in\Re^{cs}$; each $v_i\in\Re^s$ represents a prototype of class i.
- m is the weighting exponent and represents the level of fuzziness; $1\le m<\infty$.

---

[9][Ruspini1969] was the first one who suggested the structure of *fuzzy-c-partition* spaces. The fuzzy-c-means algorithm (originally ISODATA) was initially developed by [Dunn1974] and generalized by [Bezdek1973].
Dunn extended and developed the classical "within-groups sum of the squared errors" (WGSS) function to a fuzzy clustering criterion and developed the fuzzy-c-means clustering algorithm to minimize the objective function through an iterative method. Bezdek further extended the fuzzy objective function proposed by Dunn to a more genral form of fuzzy clustering criterion by introducing the weighting exponent m, $1\le m<\infty$. It turns out that Dunn's function is a special case (m=2) of an infinite family of objective functions.

• $\|x_k - v_i\|_A^2$ is an inner product induced norm on $\Re^s$.

By differentiation $J_m(U,v)$ with respect to u_{ik} where v_i is fixed and to v_i where U is fixed, we obtain

$$u_{ik} = \frac{1}{\sum_{j=1}^{c}\left[\frac{\|x_k - v_i\|^2}{\|x_k - v_j\|^2}\right]^{\frac{1}{m-1}}}$$

and

$$v_i = \frac{\sum_{k=1}^{n}(u_{ik})^m x_k}{\sum_{k=1}^{n}(u_{ik})^m}.$$

These two equations cannot be solved analytically, but approximate solutions can be obtained by an iterative procedure. The FCM uses iterative optimization of an objective function based on a weighted similarity measure between data points and cluster centers.

Step 1. Input the number of clusters, c, the weighting exponent, m, and the error tolerance, ε.
Step 2. Input the data X = { $x_1, x_2, \ldots, x_n$ }.
Step 3. Initialize the membership values $U = [u_{ik}]$.
Step 4. Calculate the new cluster centers $V^{(l)}$ by the 3rd equation.
Step 5. Update the $U^{(l)}$ by the 2nd equation.
Step 6. Return to Step 3, if $\left\|U^{(l+1)} - U^{(l)}\right\| > \varepsilon$; otherwise output U...

X: [sxn]	n: # of data elements - polygraph test sessions.
U: [cxn]	s: # of features - dimension of the samples in each cluster.
V: [sxc]	c: # of clusters

Fig.2: The iterative FCM[10] procedure

[10]See Fig.3 , the flow chart of the FCM code implemented in this project.

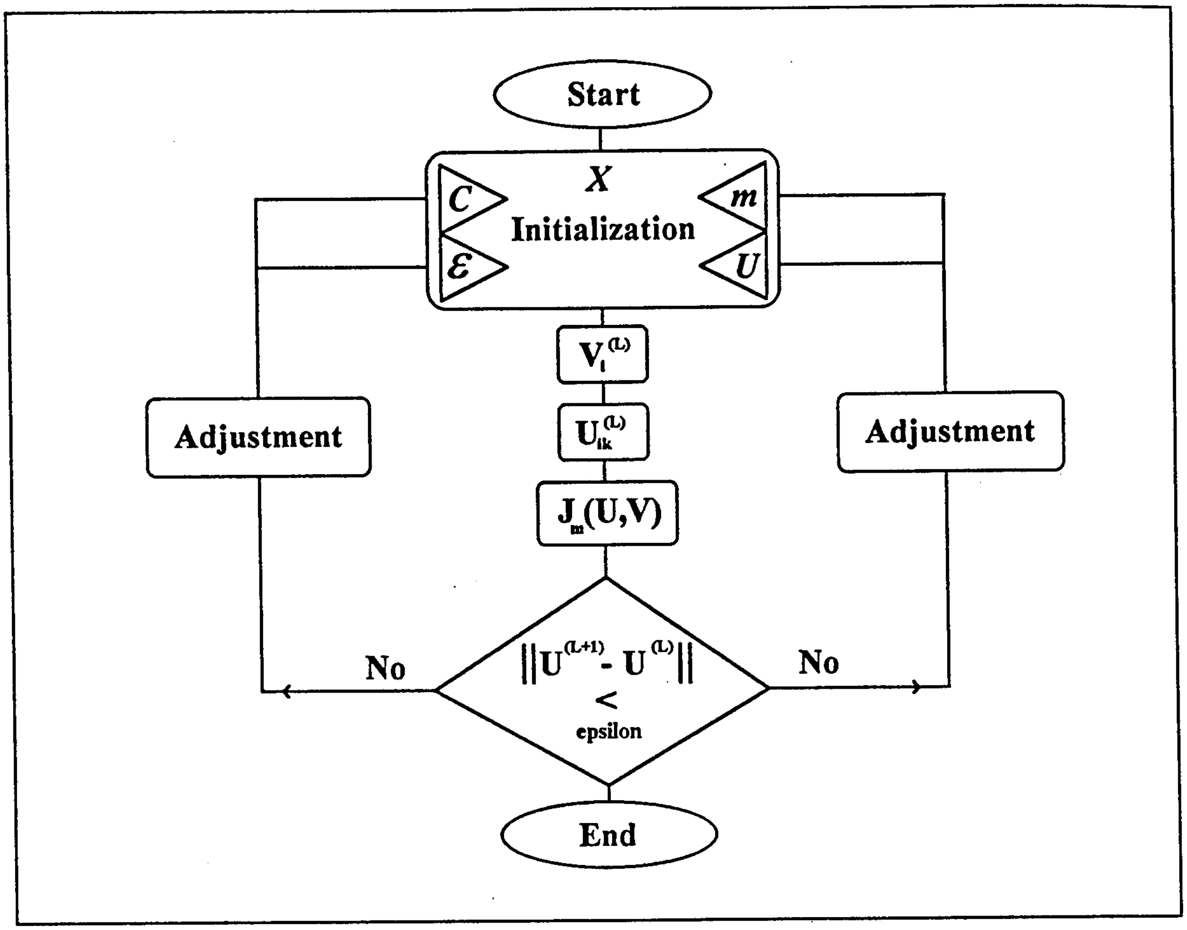


Fig.3: Flow chart of the FCM code implemented in this project

2.2.3.2. What the influential parameters practically mean or represent, and how to interpret the clustering algorithm itself:

The weighting exponent m represents the "fuzziness" level. It controls the extent of membership sharing among the fuzzy clusters. Recall the example of the two clusters, "desks" and "chairs" in chapter3.1; In a *hard* c-means clustering environment ($m \to 1$) each object can either belong to "chairs" or "desks", i.e. its membership value is either one or zero for each cluster. Now, the higher m is, the fuzzier the results will be. Thus, a desk - which can also be used as a chair- may get a membership value higher than zero for belongingness to the chairs cluster. In this sense, m controls the membership values as following

$$\lim_{m \to \infty} u_{ik} = \frac{1}{c}.$$

The control parameter epsilon represents the interrupt criterion. It influences the number of iterations and therefore the accuracy of the algorithm which is the search for c minima. By making epsilon smaller we get more accurate clustering results, but also more computing time, which is not important in this specific case.

The algorithm primarily gives us after each iteration new cluster centers V_i and new membership values U_{ik}. It then calculates the spatial distances between each data element and the found cluster centers then checks the interrupt criterion. If these distances are small enough, the algorithm will eventually give us the best membership values and the appropriate cluster centers. At this point, the search for an internal structure within the polygraph data -the original intention of every clustering process- will be finished.

FCM algorithm belongs to the so-called *partitional* clustering algorithms which generate a fuzzy c-patition matrix in a feature space. In this project I set the number of clusters c, as a known parameter, equal to two. It can otherwise be a part of the clustering optimization itself. This decision was made after running some initial tests with $c = 3$ as well, which represents "deceptive", "truthful" and "ambiguous" clusters.

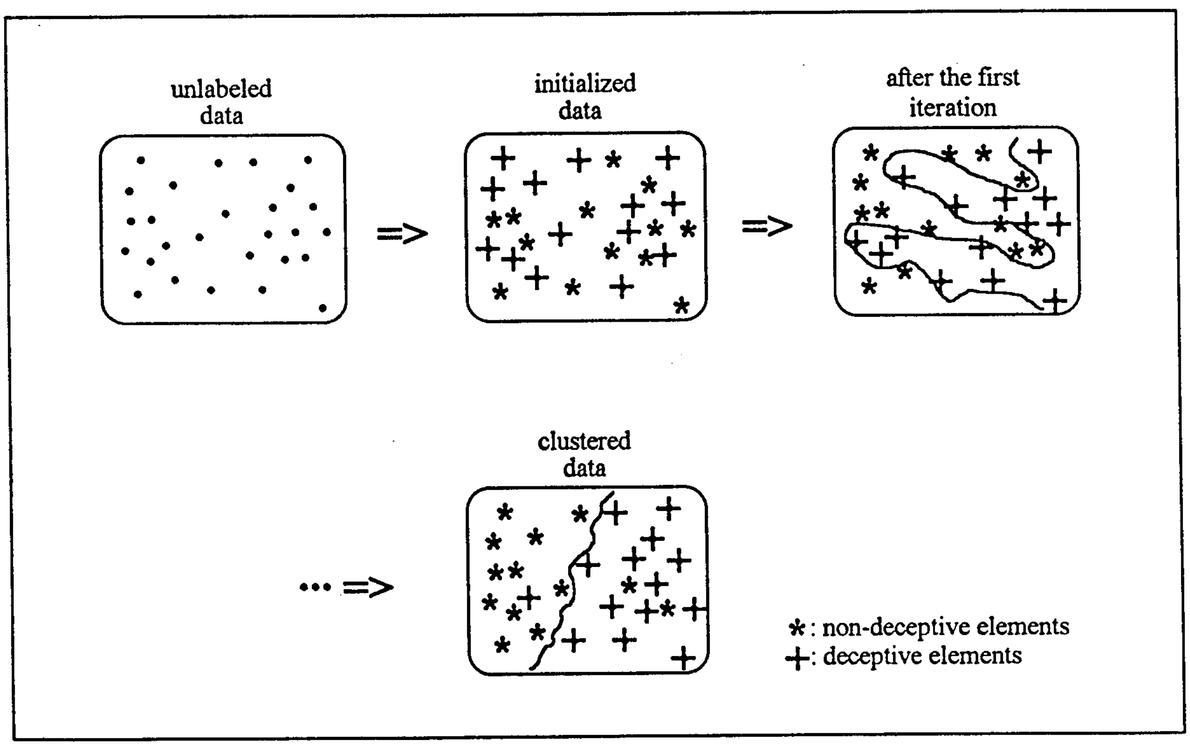


Fig.4: Fuzzy C-means algorithm applied on polygraph data

2.2.4. Why LMS fuzzy adaptive filter?

Filters are information processors. In practice, information[11] usually exists in two different modes:

- Numerical data associated with the problem,
- linguistic descriptions of human experts (often in the form of fuzzy IF-THEN rules)

Conventional filters can only process numerical data, whereas expert systems can only make use of linguistic information, i.e. a successful pattern recognition system in conventional form can only be guaranteed where either linguistic rules or numerical data do not play a critical role. Recall the fact that even in those cases we decide for a numerical method, we use linguistic information, consciously or unconsciously, in the choice among different filters, the evaluation of filter performance, the choice of the filter orders, the interpretation of filtering results, and so on.

The LMS[12] fuzzy adaptive filter is a new kind of nonlinear adaptive filter which makes use of both linguistic and numerical information concerning the physical characteristics of the polygraph data in their natural form. This filter is constructed from a set of changeable fuzzy IF-THEN rules, i.e. we have the choice of setting the rules according to our experiences and incorporating them directly into the filter, or initializing the rules arbitrarily; similar to the polynomial, neural nets, or radial basis function adaptive filters.

2.2.5. LMS fuzzy adaptive filter and its interpretation:

2.2.5.1. Filter code - An adaptive procedure

As stated before, this filter is constructed from a set of changeable fuzzy IF-THEN rules by matching input-output pairs through an adaptation procedure. The adaptive algorithm updates the parameters of the membership functions which characterize the fuzzy concepts in the IF-THEN rules by minimizing a criterion function.

Consider a real-valued vector sequence $[\underline{x}(k)]$ and a real valued scalar $[d(k)]$. The adaptive filter $f_k\colon U \rightarrow R$ is to determine, such that $L = E[(d(k) - f_k(\underline{x}(k)))^2]$ is minimized.

---

[11] About the pattern of the subject to be studied.

[12] LMS = Least Mean squares

With $k = 1,2,3,...$ and $\underline{x}(k) \in U \equiv [C_1^-, C_1^+] \times [C_2^-, C_2^+] \times \cdots \times [C_n^-, C_n^+] \subset R^n$. U and R are the input and output spaces of the filter, respectively.

The following steps describe the LMS fuzzy adaptive filter[13] used in this project:

Step 1: M fuzzy sets F_i^l are to be defined in each interval $[C_i^-, C_i^+]$ of U with the following Gaussian membership functions

$$\mu_{F_i^l}(x_i) = \exp\left[-\frac{1}{2}\left(\frac{x_i - \overline{x_i^l}}{\sigma_i^l}\right)^2\right]$$

where $l = 1,2,...,M$, $i = 1,2,...,n$, $x_i \in [C_i^-, C_i^+]$, and $\overline{x_i^l}$ and σ_i^l are free parameters which will be updated in the LMS adaptation procedure of Step 4.

Step 2: A set of M fuzzy IF-THEN rules is to be constructed in the following form:

R^l: IF x_1 is F_1^l and ... x_n is F_n^l, THEN d is G^l,

...

R^M: IF x_1 is F_1^M and ... x_n is F_n^M, THEN d is G^M.

where $\underline{x} = (x_1,...,x_n) \in U$, $d \in R$, F_i^l's are defined in Step 1, and G^l's are fuzzy sets defined in R. The (parameters of) membership functions $\mu_{F_i^l}$ and μ_{G^l} in these rules will change during the LMS adaptation procedure of step 4. Therefore, the rules constructed in this step are *initial* rules of the fuzzy adaptive filter.

Step 3: The filter $f_k: U \to R$ is constructed based on the M rules of the Step 2 as follows:

$$f_k(\underline{x}) = \frac{\sum_{l=1}^{M} \theta^l \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)}{\sum_{l=1}^{M} \left(\prod_{i=1}^{n} \mu_{F_i^l}(x_i)\right)}$$

where $\mu_{F_i^l}$'s are the Gaussian membership functions of Step 1, and $\theta^l \in R$ is any point at which μ_{G^l} achieves its maximum value.

[13]This algorithm is suggested in [Wang1993] and [Wang1994].

Because we chose the membership functions to be Gaussian functions which are nonzero for any $x_i \in [C_i^-, C_i^+]$, the denominator of the last equation is nonzero for any $\underline{x} \in U$. Therefore, the filter f_k is well defined, and because the θ^l as well as $\overline{x_i^l}$ and σ_i^l are free parameters, this filter is nonlinear in the parameters.

Step 4: The following LMS algorithm [Widrow1985] is used to update the filter parameters θ^l, $\overline{x_i^l}$ and σ_i^l. With the initial $\theta^l(0)$, $\overline{x_i^l}(0)$ and $\sigma_i^l(0)$ values determined in Step 2, the adaptive procedure is as following:

$$\theta^l(k) = \theta^l(k-1) + \alpha[d(k) - f_k]\frac{a^l(k-1)}{b(k-1)}$$

$$\overline{x_i^l}(k) = \overline{x_i^l}(k-1) + \alpha[d(k) - f_k]\frac{\theta^l(k-1) - f_k}{b(k-1)}a^l(k-1)\frac{x_i(k) - \overline{x_i^l}(k-1)}{(\sigma_i^l(k-1))^2}$$

$$\sigma_i^l(k) = \sigma_i^l(k-1) + \alpha[d(k) - f_k]\frac{\theta^l(k-1) - f_k}{b(k-1)}a^l(k-1)\frac{\left(x_i(k) - \overline{x_i^l}(k-1)\right)^2}{(\sigma_i^l(k-1))^3}$$

where $a^l(k-1) = \prod_{i=1}^{n} \exp[-\frac{1}{2}(\frac{x_i(k) - \overline{x_i^l}(k-1)}{\sigma_i^l(k-1)})^2]$, $b(k-1) = \sum_{l=1}^{M} a^l(k-1)$, $f_k = \frac{\sum_{l=1}^{M} \theta^l a^l(k-1)}{b(k-1)}$

and α is a small positive step-size. These equations are obtained by taking the gradient of L ignoring the expectation E (see chapter 2.2.5.1).

2.2.5.2. Influential parameters - meanings & interpretations:

The LMS algorithm is a gradient algorithm, i.e. a good choice of initial parameters θ^l, $\overline{x_i^l}$ and σ_i^l is very important to its convergence concerning accuracy and time. Since the error measure of this "back-propagation" algorithm is an extremely complicated function of all the parameters θ^l, $\overline{x_i^l}$ and σ_i^l, it can have numerous local minima. Depending on the initial parameter estimates, this algorithm always leads to the nearest minimum, i.e. it can become stuck in a local minimum of the error measure.

Recall that this filter is constructed based on linguistic rules from our previous experiences and some arbitrary rules. Both sets of rules are updated during the LMS adaptation procedure of Step 4 by changing the parameters in the direction of minimizing L.

In other words, the adaptation procedure can be directed to *the* local minimum we want (i.e. accuracy factor) and can converge quickly (i.e. time factor).

if these rules provide good instructions for how the filter should perform, that is, good description of the input-output pairs $[\underline{x}(k);d(k)]$.

The updating parameters θ^l[Mx1], $\overline{x_i^l}$[MxN] and σ_i^l[MxN] represent output means, input means and the input width of the Gaussian distributed data, respectively. The scalar output d is basically the label[14] of the test data[1xN] in numerical form, and σ_i^l describes how far the data from the output mean can be and still be assigned to it in an appropriate fuzzy form. M represents the number of the rules and N the number of the features, i.e. the dimension of the data. The parameter α is the "learning factor" or the step-size of training. It represents how fast and how smooth the training process proceeds.

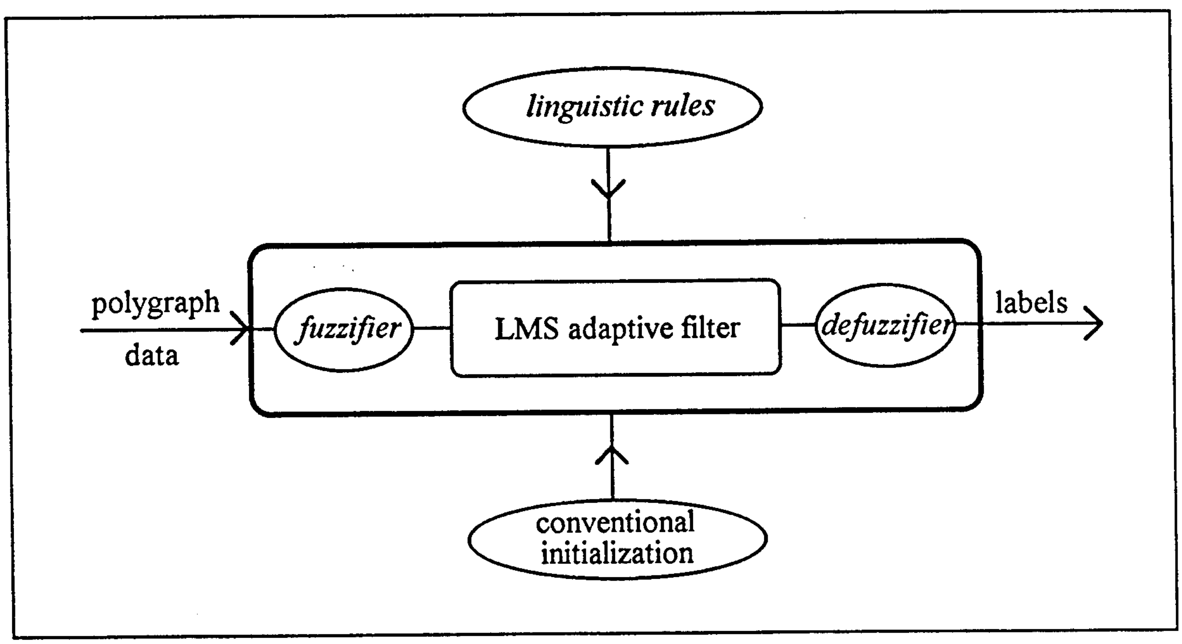


Fig.5: The LMS *fuzzy* adaptive filter used in this project

[14]"deceptive" or "non-decptive".

§3. APPROACH

3.1. Part I - FCM

3.1.2. Initial stage (conditions and methods):

A primary component of every pattern recognition problem is feature extraction. And this is actually one of the most important and influential tasks for any successful approach.

In previous researches [Layeghi1993,1] [Jacobs1993] [Dastmalchi1993], students have already investigated a set of 669 features for each polygraph test session. They used these features to train, optimize and eventually classify the data by a fuzzy K-Nearest Neighbor algorithm (KNN).

In this project, I have used these same features in their original form. I have also selected their best features and feature combinations for initial tests of my algorithm and for comparison between fuzzy-CM, fuzzy LMS adaptive filter and the fuzzy KNN approach.

At this point, the question of consistency and transferability of the features - independent of the algorithm - became more significant. It turned out to be one part of this research[15].

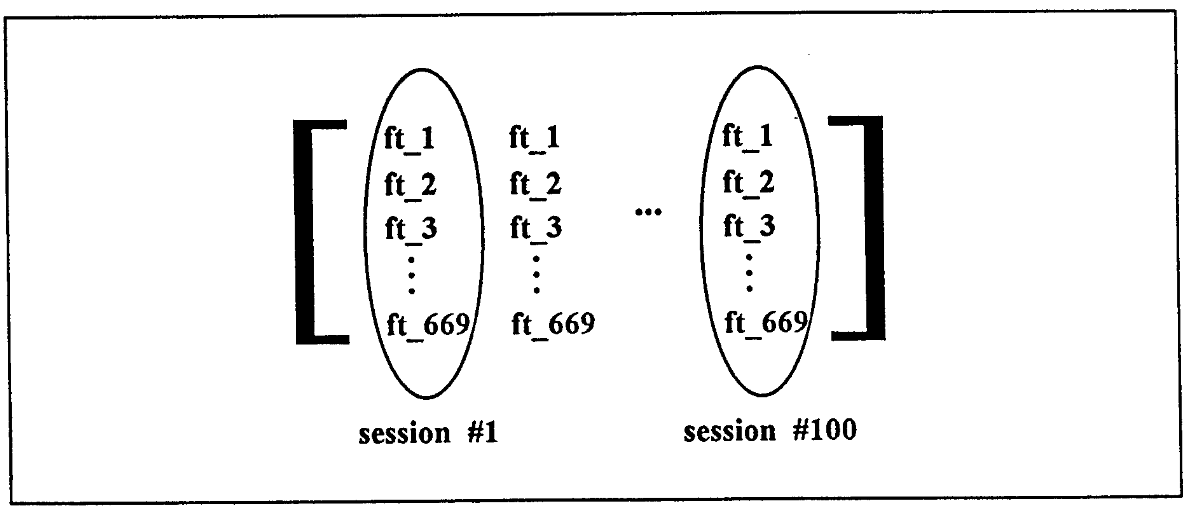


Fig.6: An example for a set of polygraph data as a matrix and its features used in this study

As mentioned earlier, each feature (total number=960) is extracted for all polygraph test *questions*, that is for relevant, irrelevant and control questions. It was, however, decided

[15]See also chapter 4.1.2.3.

not to use irrelevant questions in this study, because in a Controlled Question Polygraph Test comparison between the responses to relevant and control questions is the actual and most important factor.

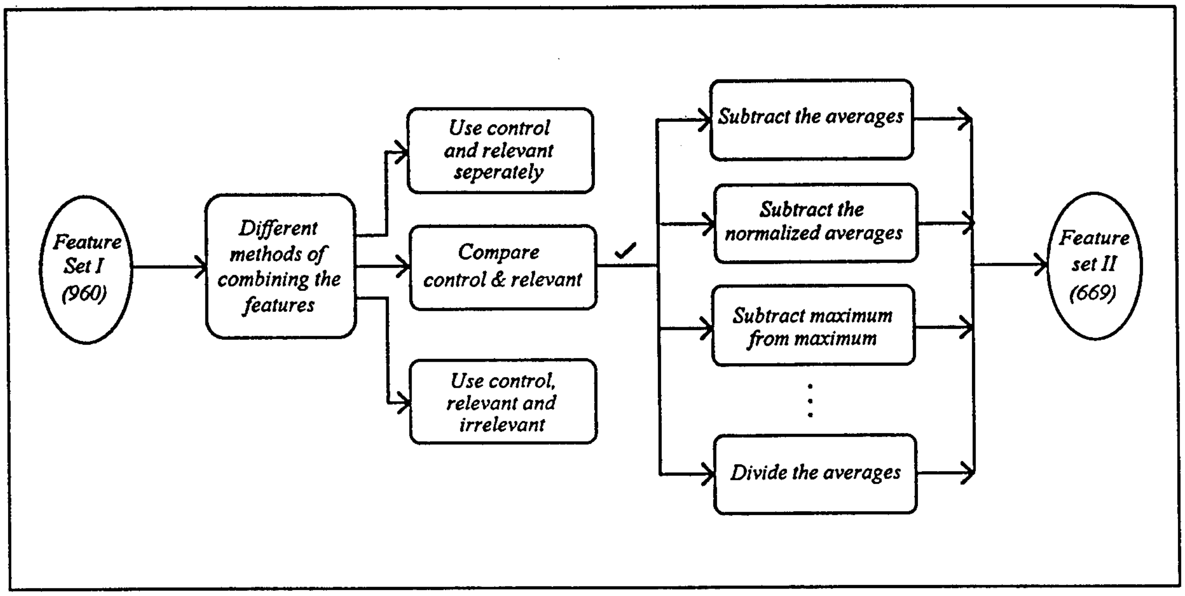


Fig.7: The original feature combinations

The Total number of the features for every test *session* at this stage is 669. Each set contains the same non-deceptive files but different deceptive ones. For more specific details about how the feature extraction was processed, and about combination methods which narrowed the total number from 960 to 669, see the references mentioned above.

3.1.3. Clustering stage

3.1.3.1. One-dimensional search and selection of the "best" single features:

After implementation and initial tests of the FCM-code, I began with the one-dimensional clustering (using *one* feature for all sessions). I used three sets (polydat_1, polydat_2, polydat_3) of such structured data as shown in Fig.42 containing 100 data elements, i.e. 50 truthful and 50 deceptive files. With these data, we ran 669 one-dimensional clustering searches containing 100 different one-dimensional data points at each time. As a result, we attained 669 times 2 clusters for each polydat_i.

After running these tests and evaluating them, I decided to select four sets of "best" one-dimensional features out of each polydat_i in preparation for the multi-dimensional clustering search. This decision was necessary to narrow the number of features, since it is impractical to find the best combination (concerning the quantity and the quality)[16] out of this immense number of features by an exhaustive way of searching.

For example, chosing only 4 or less feature-tuples from a set of 669 by trying all the possible different combinations needs the following number of computations:

$$\sum_{i=1}^{4} \binom{669}{i} = \sum_{i=1}^{4} \frac{669!}{i!(669-i)!} \approx 10^{10}.$$

The other challenge while finding good feature combinations is the problem of *single* features which yield poor results by one-dimensional clustering, but when used in *combination* with other features yield very good[17] results.

To narrow the amount of possible features, I decided to select the following four sets of single features with different performances.

	percentage of right detections in		
	deceptive files		*non-deceptive files*
group 1	≥ 60%	&	≥ 60%
group 2	≥ 80%	&	≥ 50%
group 3	≥ 50%	&	≥ 80%
group 4a	≥ 98%	&	no constraints
group 4b	no constraints	&	≥ 98%

Fig.8: Selected features by using one-dimensional FCM

The threshold of 60% was chosen, because any other value below or above that limit would again give us either too many or not enough features. Furthermore, any other value

[16]That means: How many features and which ones should be taken in a combination.
[17]"Good" or "poor" in sense of the definition in chapter 1.1.2.

closer to the limit 50% for both deceptive and non-deceptive files would be only a *random* clustering process. Yet, this decision was not enough. We would have lost some good features which provide correct detections - better than 80% - for at least one of the files. The fourth group was chosen to enable us to consider some extreme cases.

As an additional set of one-dimensional features, I chose those with good results in multi-dimensional tests[18] for *one* of the polydat_i's, and used them also for the other two polydat_i's, even though they didn't belong to one of the four feature sets mentioned above. This set was important to fulfill the constraint of consistency and transferability for any chosen polygraph data[19].

---

[18]See chapter 3.1.3.2.
[19]See the comparison in chapter 4.1.2.3.

Σ=669

ft_#	w-dcp #	dcp-ok %	w-non #	non-ok %	iter_#
1.0000	12.0000	76.0000	9.0000	82.0000	13.0000
2.0000	37.0000	26.0000	44.0000	12.0000	15.0000
3.0000	16.0000	68.0000	10.0000	80.0000	14.0000
4.0000	12.0000	76.0000	18.0000	64.0000	15.0000
5.0000	15.0000	70.0000	16.0000	68.0000	16.0000
6.0000	38.0000	24.0000	27.0000	46.0000	15.0000
7.0000	48.0000	4.0000	0	100.000	40.0000
8.0000	22.0000	56.0000	9.0000	82.0000	8.0000
9.0000	22.0000	56.0000	8.0000	84.0000	13.0000
10.0000	22.0000	56.0000	11.0000	78.0000	38.0000
11.0000	0	100.000	33.0000	34.0000	26.0000
12.0000	20.0000	60.0000	15.0000	70.0000	6.0000
13.0000	46.0000	8.0000	26.0000	48.0000	10.0000
14.0000	22.0000	56.0000	11.0000	78.0000	16.0000
15.0000	12.0000	76.0000	9.0000	82.0000	27.0000
16.0000	37.0000	26.0000	44.0000	12.0000	17.0000
17.0000	16.0000	68.0000	10.0000	80.0000	25.0000
18.0000	12.0000	76.0000	17.0000	66.0000	37.0000
19.0000	15.0000	70.0000	16.0000	68.0000	40.0000
20.0000	38.0000	24.0000	27.0000	46.0000	34.0000
21.0000	48.0000	4.0000	0	100.000	31.0000
22.0000	12.0000	76.0000	14.0000	72.0000	25.0000
23.0000	10.0000	80.0000	45.0000	10.0000	20.0000
24.0000	21.0000	58.0000	15.0000	70.0000	23.0000
25.0000	18.0000	64.0000	24.0000	52.0000	29.0000
26.0000	24.0000	52.0000	19.0000	62.0000	18.0000
27.0000	12.0000	76.0000	23.0000	54.0000	22.0000
28.0000	46.0000	8.0000	2.0000	96.0000	35.0000
29.0000	18.0000	64.0000	9.0000	82.0000	28.0000
30.0000	12.0000	76.0000	10.0000	80.0000	14.0000
		...			
447.0000	17.0000	66.0000	36.0000	28.0000	17.0000
448.0000	7.0000	86.0000	40.0000	20.0000	25.0000
449.0000	16.0000	68.0000	11.0000	78.0000	15.0000
450.0000	12.0000	76.0000	9.0000	82.0000	15.0000
451.0000	13.0000	74.0000	18.0000	64.0000	20.0000
452.0000	5.0000	90.0000	20.0000	60.0000	13.0000
453.0000	18.0000	64.0000	18.0000	64.0000	12.0000
		...			
662.0000	27.0000	46.0000	34.0000	32.0000	9.0000
663.0000	16.0000	68.0000	30.0000	40.0000	9.0000
664.0000	21.0000	58.0000	37.0000	26.0000	17.0000
665.0000	31.0000	38.0000	23.0000	54.0000	14.0000
666.0000	34.0000	32.0000	17.0000	66.0000	45.0000
667.0000	25.0000	50.0000	28.0000	44.0000	20.0000
668.0000	15.0000	70.0000	37.0000	26.0000	12.0000
669.0000	15.0000	70.0000	39.0000	22.0000	11.0000

Feature number: **ft_#**
# of wrong results in decept. data: **w-dcp**
% right detection in decept. data: **dcp-ok**
# of wrong results in truthful data: **w-non**
% right detection in truthful data: **non-ok**
Iterations_# for each feature: **iter_#**

Fig.9: An example for one-dimensional clustering

ft_#	w-dcp #	dcp-ok %	w-non #	non-ok %	iter_#
1.0000	12.0000	76.0000	9.0000	82.0000	13.0000
3.0000	16.0000	68.0000	10.0000	80.0000	14.0000
4.0000	12.0000	76.0000	18.0000	64.0000	15.0000
5.0000	15.0000	70.0000	16.0000	68.0000	16.0000
12.0000	20.0000	60.0000	15.0000	70.0000	6.0000
15.0000	12.0000	76.0000	9.0000	82.0000	27.0000
17.0000	16.0000	68.0000	10.0000	80.0000	25.0000
18.0000	12.0000	76.0000	17.0000	66.0000	37.0000
19.0000	15.0000	70.0000	16.0000	68.0000	40.0000
22.0000	12.0000	76.0000	14.0000	72.0000	25.0000
29.0000	18.0000	64.0000	9.0000	82.0000	28.0000
30.0000	12.0000	76.0000	10.0000	80.0000	14.0000
31.0000	14.0000	72.0000	16.0000	68.0000	21.0000
33.0000	18.0000	64.0000	16.0000	68.0000	14.0000
36.0000	15.0000	70.0000	8.0000	84.0000	14.0000
37.0000	8.0000	84.0000	13.0000	74.0000	15.0000
38.0000	12.0000	76.0000	14.0000	72.0000	18.0000
39.0000	14.0000	72.0000	13.0000	74.0000	17.0000
40.0000	16.0000	68.0000	15.0000	70.0000	13.0000
50.0000	17.0000	66.0000	17.0000	66.0000	18.0000
52.0000	15.0000	70.0000	20.0000	60.0000	23.0000
68.0000	13.0000	74.0000	18.0000	64.0000	17.0000
70.0000	20.0000	60.0000	20.0000	60.0000	23.0000
82.0000	16.0000	68.0000	20.0000	60.0000	12.0000
141.0000	17.0000	66.0000	17.0000	66.0000	15.0000
155.0000	17.0000	66.0000	17.0000	66.0000	25.0000
176.0000	16.0000	68.0000	18.0000	64.0000	13.0000
177.0000	16.0000	68.0000	16.0000	68.0000	13.0000
197.0000	13.0000	74.0000	17.0000	66.0000	15.0000
200.0000	17.0000	66.0000	13.0000	74.0000	12.0000
211.0000	13.0000	74.0000	16.0000	68.0000	42.0000
214.0000	17.0000	66.0000	12.0000	76.0000	27.0000
216.0000	15.0000	70.0000	14.0000	72.0000	32.0000
235.0000	15.0000	70.0000	19.0000	62.0000	14.0000
395.0000	18.0000	64.0000	17.0000	66.0000	10.0000
449.0000	16.0000	68.0000	11.0000	78.0000	15.0000
450.0000	12.0000	76.0000	9.0000	82.0000	15.0000
451.0000	13.0000	74.0000	18.0000	64.0000	20.0000
452.0000	5.0000	90.0000	20.0000	60.0000	13.0000
453.0000	18.0000	64.0000	18.0000	64.0000	12.0000
458.0000	16.0000	68.0000	14.0000	72.0000	8.0000
459.0000	20.0000	60.0000	10.0000	80.0000	10.0000
460.0000	14.0000	72.0000	18.0000	64.0000	9.0000
462.0000	14.0000	72.0000	17.0000	66.0000	7.0000
600.0000	18.0000	64.0000	20.0000	60.0000	37.0000

$\Sigma=45$

Feature number: ft_#
\# of wrong results in decept. data: w-dcp
% right detection in decept. data: dcp-ok
\# of wrong results in truthful data: w-non
% right detection in truthful data: non-ok
Iterations_# for each feature: iter_#

Fig.10: An exmple for the first group of selected features (representing group #1 at page)

3.1.3.2. Multi-dimensional search for the best feature combination:

3.1.3.2.1. Overview:

Having obtained these four sets of features, a multi-dimensional searching process through all of them was initiated to find the best feature combinations (concerning the quantity and the quality[20]).
Even though the number of the features[21] has already been narrowed, it is still impractical to do an exhaustive search, since the total number of the features contained in these four sets is about 100 for each polydat_i. In other words, the following number of computations is still needed for calculation of all 4 or less possible feature-tuples:

$$\sum_{i=1}^{4} \binom{100}{i} = \sum_{i=1}^{4} \frac{100!}{i!(100-i)!} \approx 4.0 \cdot 10^6 .$$

At this stage, I decided to investigate 3 different search methods to bypass the exhaustive way. They are

1. *random* search without duplication of any feature within a tuple,
2. *pseudo-exhaustive* search with the option of duplication and finally
3. *genetic* search with "uncontrollable" possibility of duplications.

In previous research projects [Layeghi1993,1] [Dastmalchi1993] [Jacobs1993], it was decided to narrow the feature numbers from 669 to 30 "best" ones and then an exhaustive search was run for up to four- or five-tuple combinations. In other words, their strategy was completely different than the aforementioned three strategies.

As mentioned before a "poor" or an average *single* feature by one-dimensional clustering might give us in *combination* with other features very good or even better results by a multi-dimensional clustering than any of them individually.
This fact was totally neglected by the feature selection methods used in the previous researches[22] [Laueghi1993,1] [Dastmalchi1993].

---

[20]That means: How many features and which ones should be taken in a combination.
[21]See chapter 3.1.3.1.
[22]See chapter 4.3. comparison for more details about differences between this and previous works.

Applying these three new strategies, I was able to consider more possible features for a multi-dimensional clustering than in previous works, without using the impractical exhaustive method.

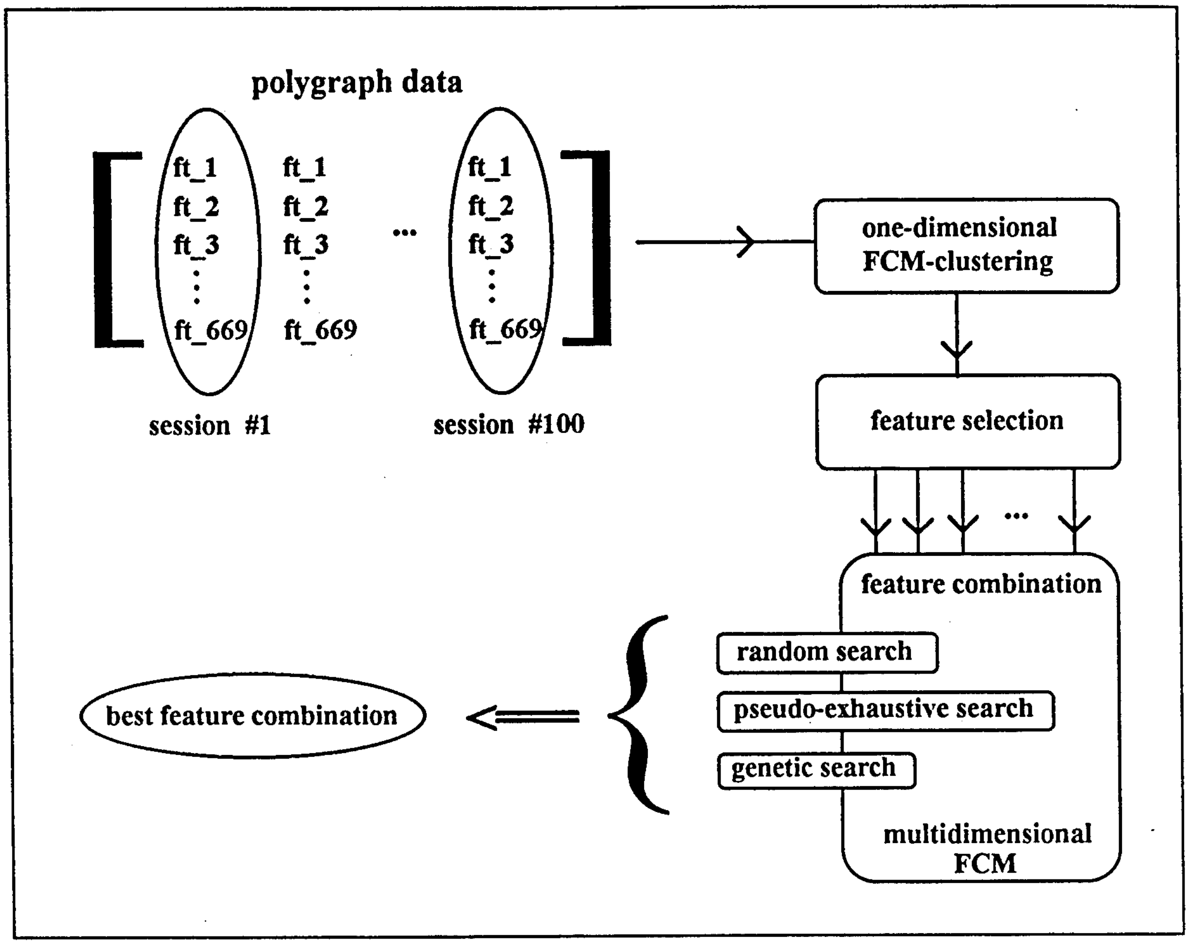


Fig.11: General search to find the best feature combination

3.1.3.2.2. Random search method:

Applying this method, an average of 14 to 20 different features out of the aforementioned four sets were taken, and then the FCM algorithm including the evaluation program for randomly chosen 4-tuples were run. After about 1000 combinations were constructed, I then picked out the best features and their combinations, and replaced the poor ones with new features. This same procedure was repeated until good[23] combinations were found.

[23]"Good" in sense of the definition in chapter 1.1.2.

Every time the results were out of balance - i.e. highly better detection either for deceptive or non-deceptive files by the cost of the other one - I appropriately took additional features from those four sets to eliminate the difference by improving the results of the worse file - and as much as possible - by maintaining the results of the better file.

After running this kind of tests several times, we were able to estimate which features are the good ones to combine together.

3.1.3.2.3. Pseudo-exhaustive search method:

Having some idea[24] which features are good in a combination with others[25], I built *every* possible four- to six-tuples out of those features and evaluated them. This method was very important to make sure that we did not lose any good combinations which might have been neglected by the random search.

I called this method *"pseudo"*-exhaustive, because each time it considers only a small part of the available features; but *"exhaustive"*, because it takes all the possible combinations within this part. Except for this major difference, all the other steps of this method are exactly the same as the random search.

3.1.3.2.4. Genetic search method:

This algorithm is basically a compromise between the pseudo-exhaustive and the random search method, plus a weighting system which supports those features with good results.

Initial populations of 200 to 300 chromosomes[26] are randomly created. Each chromosome is a combination of N features, where N stays constant for each population during the outgrowth. Each single feature is selected from a gene pool for the particular population that the individual belongs to. Each gene pool consists of twenty to forty features that we have chosen[27].

---

[24]By using the results of the random search method and also the 5th group mentioned at page 3.1.3.1.
[25]Remember the fact that some "poor" single features might give us in combination with others very good results
[26]Individuals or feature-tuples.
[27]Directed by our experience from using the random and the pseudo-exhaustive methods.

In this project three processes operate on the evolution[28] of each population:

- reproduction
- crossover
- mutation.

These three processes determine how each new generation will be created based on the old one. Before genetic reproduction, the fuzzy-c-means algorithm evaluates the percentage of correct deceptive and non-deceptive detections for each chromosome. The average of them is the fitness value of that chromosome. During the genetic reproduction, the chromosomes of the new generation are copied from the chromosomes of the old generation in a probabilistic sense. The probability that a particular chromosome will be copied is the ratio of that chromosome's fitness value against the total fitness values of the entire population of the old generation.

After selection, genetic crossover randomly chooses pairs of chromosomes as parents, splices them, and recombines them - by randomly mixing some of the parents genes - into pairs of offsprings. Finally, genetic mutation randomly substitutes a new gene within a randomly chosen chromosome. The extent to which crossover and mutation occur can be verified by appropriate initialization.

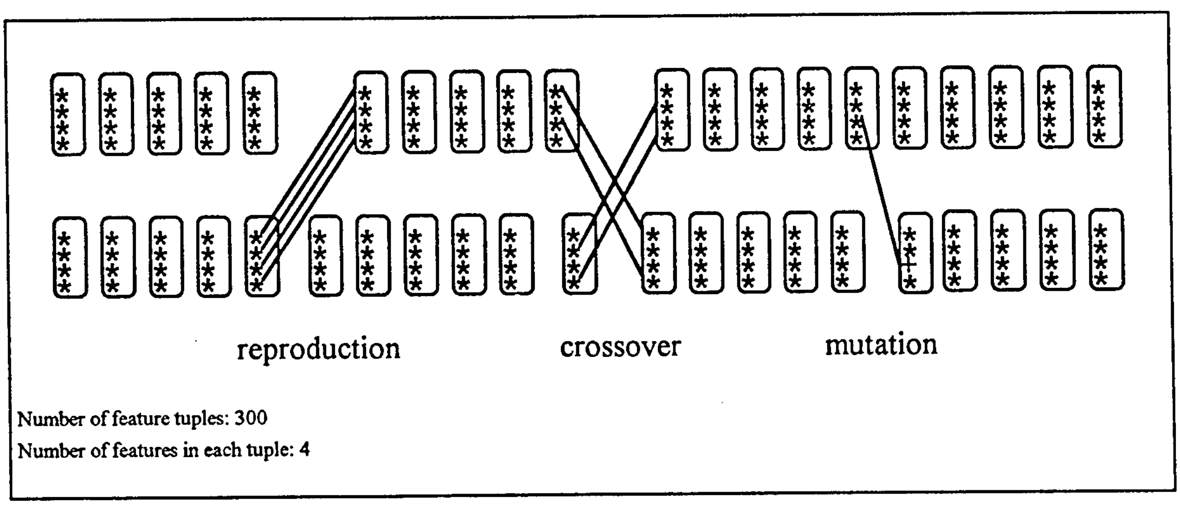


Fig.12: An example for the genetic outgrowth with 4 genes (=features) in each chromosome (=individual)

[28]See chapter 4.1.2.2 for particular results of this method.

3.1.3.3. General process - Optimization by changing parameters:

Simultaneously to the search for the best features and their combinations, we were optimizing the system by changing and adjusting the parameters. Recall, the whole idea of this pattern recognition was to cluster the unlabeled data into two clusters which represent the deceptive and the truthful group[29].

Knowing the information of which files were deceptive or truthful[30], we were able to change the parameters in the way that the output could continuously come closer to the real cluster structure. This process is depicted in the following figure. The "fuzzy c-means algorithm" block not only represents the pure FCM algorithm shown in Fig.3, but also the general search for good features shown in Fig.11 which ran simultaneously with the optimization process.

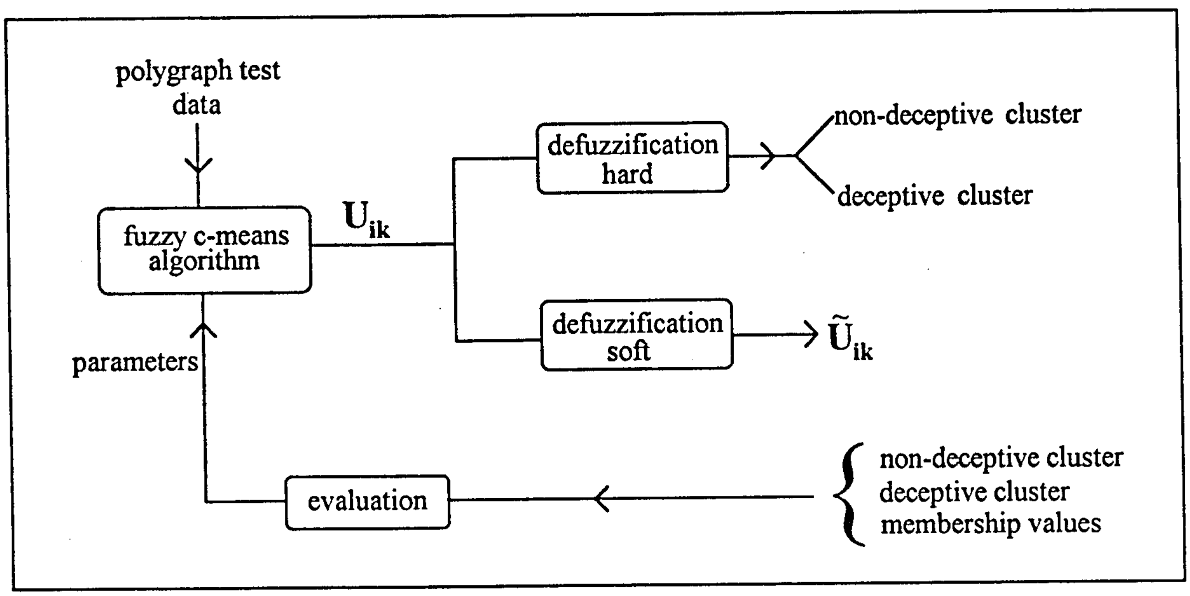


Fig.13: Optimization of the clustering environment
- General process -

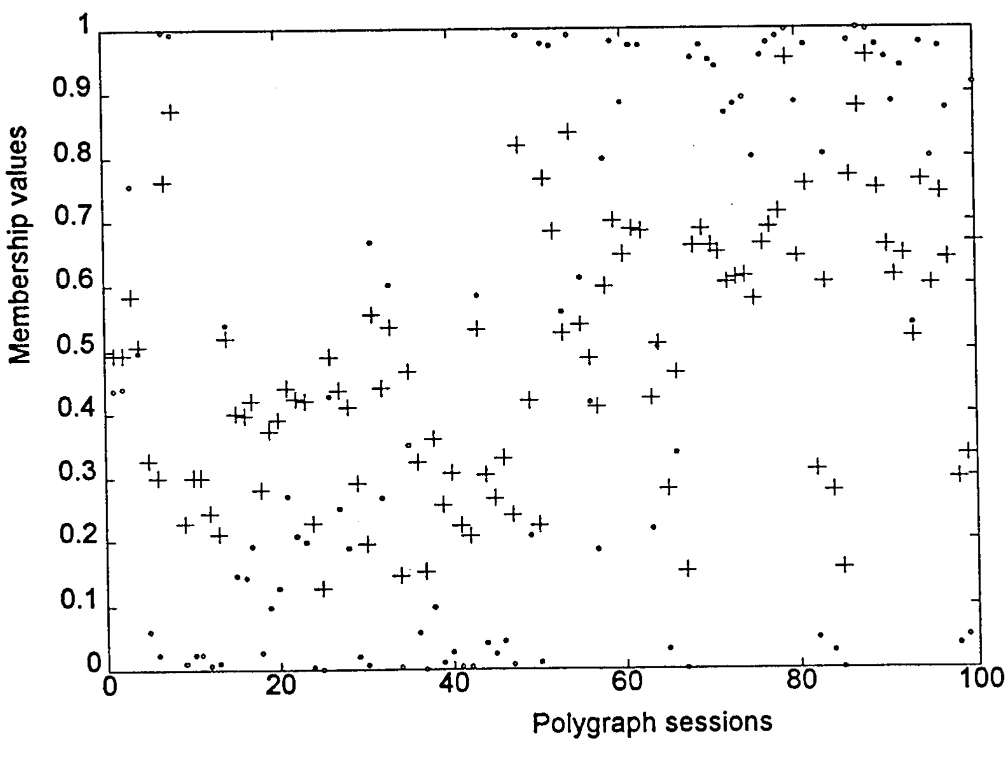
As an example, I will briefly discuss how the parameter m was chosen and eventually modified: The weighting exponent m plays a significant role in this system. Since the control parameter m itself does not belong to the optimizing values within the iterative process of FCM algorithm, one must choose m before implementing the algorithm, and

[29]See chapter 3.1.2.
[30]We know this information beforehand for sure, because the subjects have confessed their case or the actual offender was found.

optimize it manually. There are several research papers written as an attempt to find the optimal m for different clustering problems.

The effect of m was discussed in [Bezdek1981]. Although Bezdek proposed heuristic guidelines for m, no *theoretical* basis for an optimal choice for *m* has been reported. The only known paper in this matter [Choe1992] proposed a method for determining m based on the concept of fuzzy decision theory initiated by [Zadeh1970].

But since the definition of "good" clusters in [Choe1992] did not exactly match to our clustering environment, I chose the "trail and error" strategy to find the optimal m by systematically increasing it. Fortunately, there is a logical limit[31] for this increasing process in our case, even though m can mathematically be any value from [2, ∞).



"." represents the mmebership values for m=2
"+" represents the membership values for m=5

Fig.14: An example for the influence of 'm'

[31]See chapter 2.2.3.2. for the meaning of *m*.

For more details on this matter see the chapter 4.1.1. In Fig.14, you see an example for how the weighting exponent m influences the membership values for one of the features from polydat_3 in one-dimensional mode.

3.1.3.4. Evaluation strategy:

Due to the small number of non-deceptive cases available, each session for a subject was used as a separate and individual case. But in average, each group of three sessions belong to one person concerning the same crime, meaning the results of these sessions are not independent of each other. Using this additional information, the clustering system can come closer to the actual structure of the data, i.e. we can get a better performance.

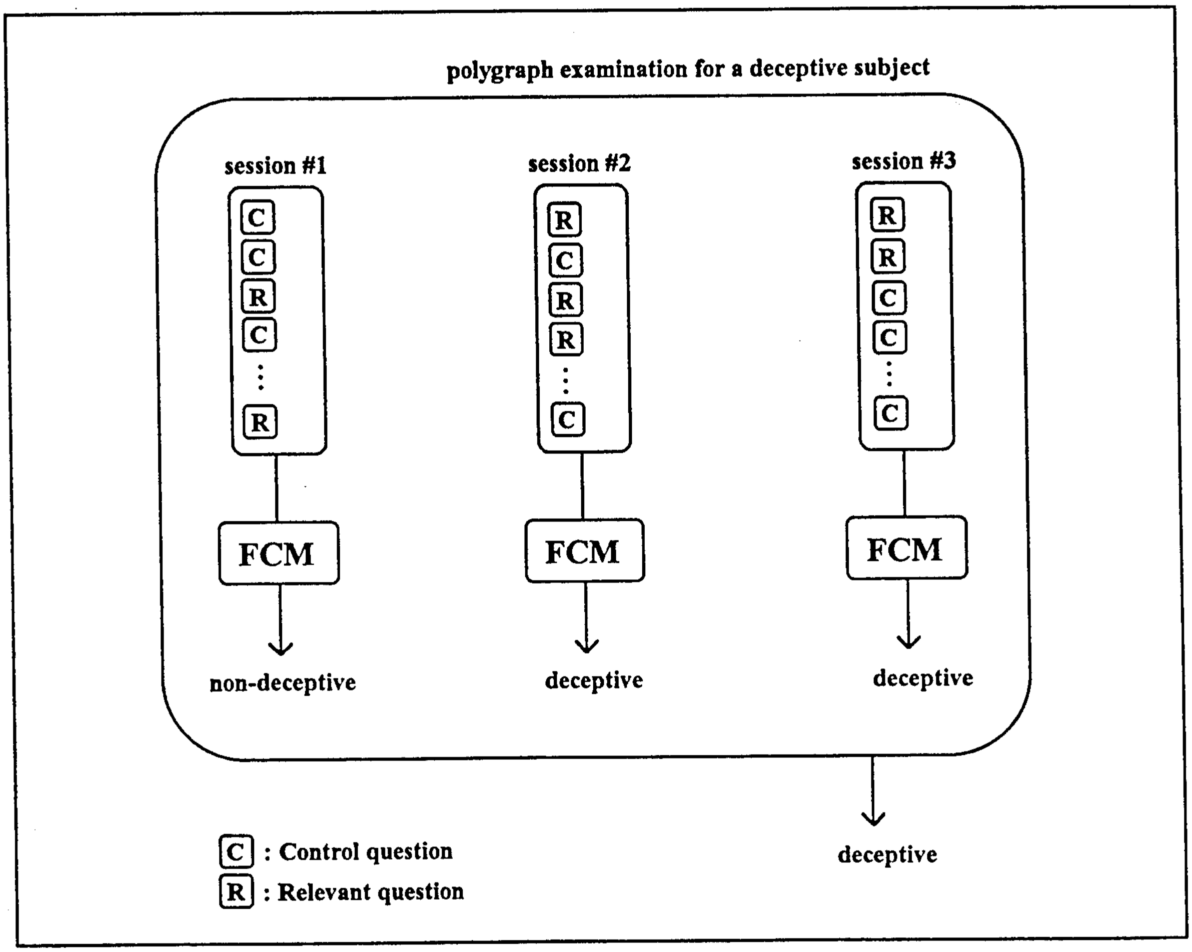


Fig.15: An example for the final evaluation using the dependency of the sessions

After clustering and evaluating[32] each session separately, some cases with different responses to the algorithm were found, although they belonged to one person. In circumstances like this, we combined the individual results within each group in a way that the majority response was assigned to the whole group (see Fig.15).

In those cases that each polygraph examination contains 2 or 4 test sessions where there is no majority response to build, I decided to take only those membership values further to the threshold 0.5. For example, by the feature combination [30, 30, 39, 235, 363, 450] used to cluster polydat_1, we obtained for one of the examination with four sessions the following membership values: 0.4164, 0.5519, 0.5377, 0.4780. After defuzzification we got 0, 1, 1, 0 where no majority class can be build. However, the second and the third membership values are closer to the threshold than the other two ones. With the aforementioned strategy, this examination is labeled with 0.

Recall that each polygraph examination has a set of control and relevant questions which is repeated an average of three times. The only difference between each session is the order in which the questions are asked.

[32] The general evaluation process is contructed as following:
After each clustering procedure (one- or multi-dimensional) a two-row vector of membership values is given which represent the two deceptive and non-deceptive clusters. The evaluation process takes the membership values of one these clusters and counts the values below and above the threshold 0.5. Thus, as a result we get the absolute number of wrong and right detections.

3.2. Part II - LMS fuzzy adaptive filter

3.2.1. Feature selection by visual inspection:

One advantage of a fuzzy logic system is its use of common sense human reasoning as inference rules. The fuzzy LMS algorithm we used extends this advantage by further optimizing such inference rules to "fit" a given set of data. To fully utilize the advantages of this fuzzy LMS algorithm, we had to face two issues: coming up with the proper intuitive rules for initialization and a set of data that reflects real-world examples for training.

As mentioned before, for practical reasons, the polygraph recognizer can use only a subset of the given 669 features, and we would have to choose the effective ones. Furthermore, the fuzzy logic system needed reasoning rules, operating on those features we selected, to analyze the data. We believed that we could visually inspect graphical plots of the feature data to learn about the feature information. Since fuzzy logic corresponds closely with human reasoning, we would then, based on the knowledge obtained from our visual inspection, select features that help differentiate deceptive and non-deceptive subjects and codify the patterns we would find into reasoning rules.

For the visual inspection, a scatter plot was made of the data in polydat_3 of each single feature. We looked at each plot individually. In any given plot, if the deceptive and non-deceptive subjects showed distinctive clusters, then the feature was considered good. If the elements of these two classes seemed to be randomly located, then the feature was considered bad. After viewing all 669 plots, we subjectively determined the following features[33] to be very good: 9, 11, 29, 164, 399, 449, 450, 451, 452, and 454; with 451 and 452 to be the best.

Initially the fuzzy adaptive filter was to be designed based on two features, with more features to be added in the future as the project progresses. We limited the feature couple to be composed of good features from the above list. Visual inspection was made of the scatter plots of the data in polydat_3 of various such feature combinations to determine the effective ones. While selecting feature couples, we again searched for combinations that show distinctive clusters for deceptive and non-deceptive subjects. The features

[33]See Fig.41 for the meaning of these numbers.

within a combination should also be uncorrelated with each other. A plot of the feature 449 and 450 combination shows that they are a bad couple because they seem to be linearly correlated[34], as the data points fall closely along a straight line.

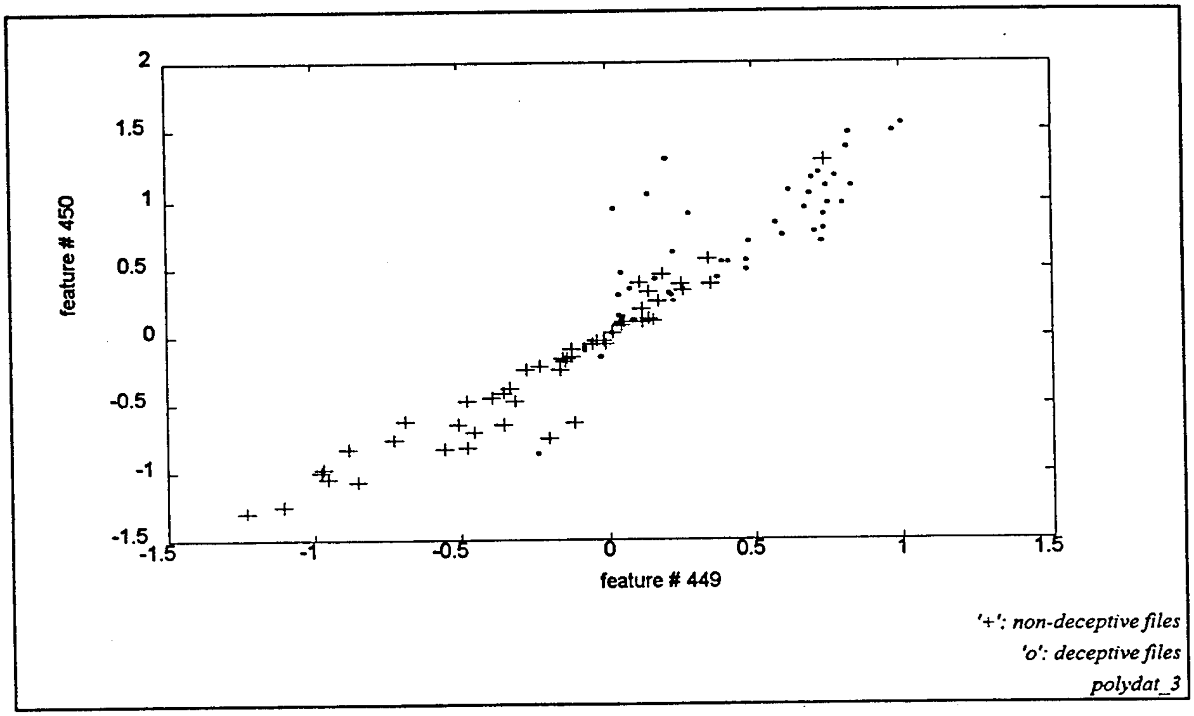


Fig.16: Scatter plots of two linearly correlated features

Visual inspection of feature couples consumed much more time than visual inspection of individual features, as the clusters took on more complicated shapes. Furthermore, in the fuzzy LMS algorithm each inference rule exerts influence centered in an elliptical contour where the major and minor axes are parallel with the axes of the feature plot. Clusters with a complicated shape must be built from those elliptical regions (see next figure). Therefore we had the additional task of finding clusters in the feature plots that could be easily approximated with few ellipses, to reduce system complexity.

Due to the lack of time, we did not examine the plots of all forty-five possible combinations of the ten very good features listed above. We only examined a random few. Based on the ones we did examine, we settled on the combination of features 451 and 452 because:

---

[34]Correlation between two features means that information in one is similar to the information in the other one, and using them together only introduces redundancy and hardly improves the system.

- they were the best - visually recognizable - features individually,
- they seemed uncorrelated with each other and
- we roughly found four elliptical clusters from the plot.

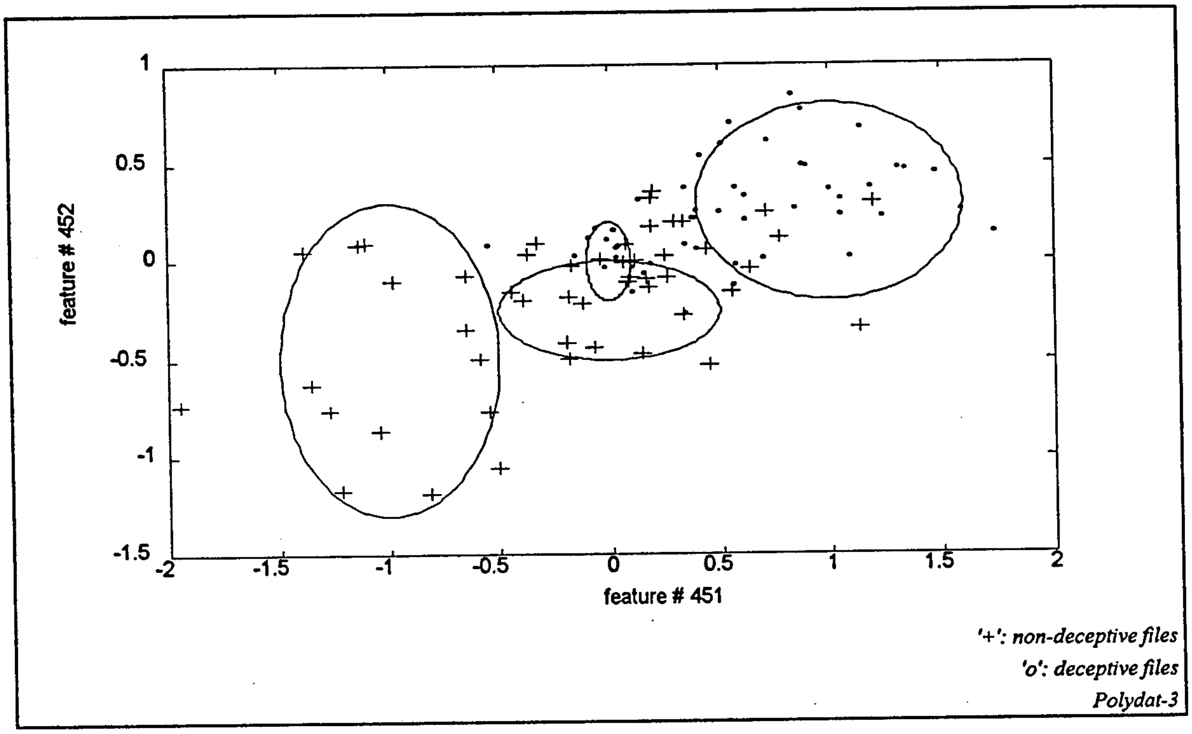


Fig.17: The four elliptical clusters used for setting the linguistic rules

3.2.2. Setting linguistic rules:

We initialized the fuzzy system such that it would exploit the knowledge we had just obtained about the clusters for features 451 and 452. There were two inputs, one for each feature, and four rules, one for each cluster. We had to represent those visual clusters we found with inference rules. The linguistic rules are shown in the following figure.

1. IF f1 is about -1 (±0.5) and f2 is about -0.5 (±0.8), THEN decision is *non-deceptive* ⇒ output is +1.
2. IF f1 is about 0 (±0.5) and f2 is about -0.25 (±0.25), THEN decision is *non-deceptive* ⇒ output is +1.
3. IF f1 is about 0 (±0.1) and f2 is about 0 (±0.2), THEN decision is *deceptive* ⇒ output is -1.
4. IF f1 is about 1 (±0.6) and f2 is about 0.3 (±0.5), THEN decision is *deceptive* ⇒ output is -1.

f1: measurement of feature # 451
f2: measurement of feature # 452

Fig.18: Initial linguistic rules for the fuzzy adaptive filter based on the clusters in Fig.17

The linguistic rules above were then translated to fuzzy membership functions as outlined in [Wang1994]. The xi's were the centers of the clusters; the sigmas were the widths of the clusters (±xxx in the above rules); and the thetas were either +1 or -1 for non-deception and deception, respectively.

The output of the fuzzy reasoning based on the above four rules would not be exactly +1 or -1. It would be within the range limited[35] by +1 and -1. For our project, we decided that a positive output denotes non-deception and a negative output denotes deception. In other words, the decision threshold was at zero.

[35] After training the output may go beyond that range.

For future investigations one may experiment with a different threshold[36].

The choice of plus and minus one for non-deception and deception is based on the following argument: The learning technique uses the squared error, which is the square of the difference between the desired output and actual output. In computing that squared error, if the difference between the desired output and actual output is greater than one, then the squaring operation expands the error value and therefore gives more significance to such mistakes. On the other hand, if the difference is less than one, than the squaring operation compresses the error value and therefore gives it less significance.

Given zero as the threshold between deception and non-deception and assuming the actual output would never go beyond plus two or minus two, then the choice of plus and minus one as desired outputs would mean that the error calculation gives more significance to misclassifications and less to correct classifications; Here classification refers to the crisp, defuzzified classification, not the degree of belonging.

For example, the desired output for non-deceptive subjects is plus one. If the actual output is between zero and two, then the crisp classification is non-deception, which is correct. The numerical difference between the actual output and the desired output is less than one in this case, and the squaring operation would lessen the significance of that error. On the other hand, if the actual output is less than zero, then the crisp classification would be deception, which is wrong. In that case, the numerical difference between the desired output and the actual output is greater than one and more significance would be given to such mistakes. Similar argument can be apply for the choice of minus one as the desired output for deceptive subjects.

3.2.3. Training, testing and evaluation strategy:

The fuzzy LMS algorithm can be optimized to a specific set of data. To exploit that aspect of the algorithm, we also selected a set of data to train the system. Following a procedure similar to one used in an earlier project with KNN classifying algorithm [Layeghi1993], we had 35 deceptive subjects and 35 non-deceptive subjects - from each polydat_i - for

[36]One may also view the output as a fuzzy value and map it to a confidence value in addition to just a deception/non-deception decision. That would differentiate a sure judgment from an unsure one and may be more helpful in practice.

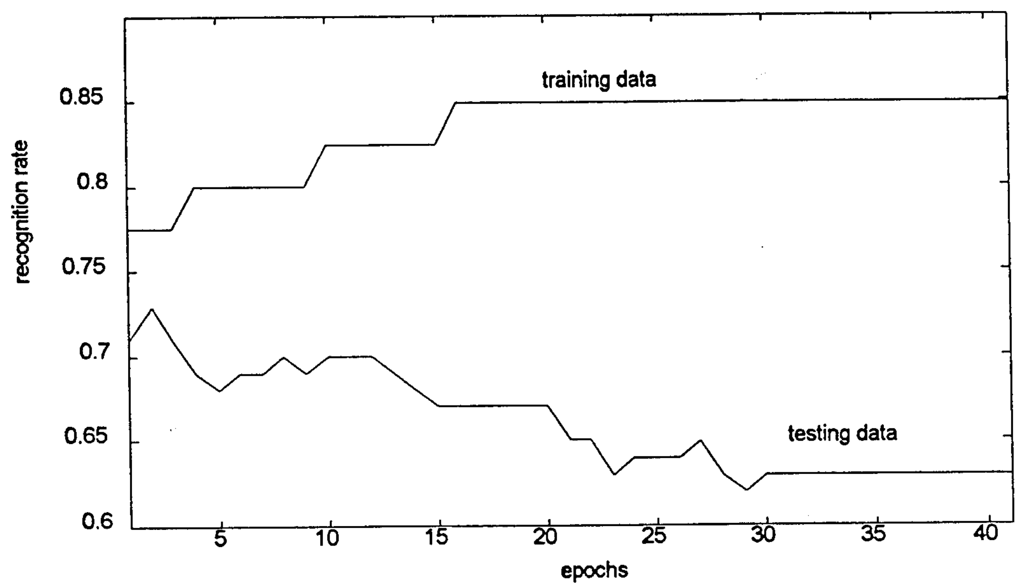
training. However, with a set of only 100 subjects within each polydat_i, that left a rather small amount for *testing* (i.e. 15 deceptive and 15 non-deceptive subjects). Therefore we also tested the algorithm with 10 deceptive subjects and 10 non-deceptive subjects for training and the rest (40 deceptive subjects and 40 non-deceptive subjects) for testing. That might be a bit extreme in the other direction, but we could interpolate the results and also see the sensitivity of the algorithm to the amount of training data.

We tested both cases for all three polydat_i's, giving a total of six tests. Each test was repeated twenty times. The training data were randomly chosen each time, and the rest of the available data in each set were used for testing. We recorded for each test the average of those twenty trials. This repeated testing was done to ensure that the results were not dependent on a particular choice of training data.

3.2.4. What to do with the memorizing problem?

Most learning algorithms suffer the dilemma of overlearning, or memorizing. Usually the problem occurs when the learning algorithm tries too hard at optimizing itself to a set of training data, sometimes to the point of memorizing them, such that it does not generalize to understand new data. Overlearning is exacerbated when the training data set is not completely representative of the testing set.

In a pattern recognition problem, while the recognition rate for the training data may increase steadily until it reaches a certain plateau, the recognition rate for testing data may only increase for a while, after which it may decrease until it hits a plane. We observed such phenomenon in our system:



The training data consist of twenty non-deceptive subjects and twenty deceptive subjects from polydat_3.
The testing data consist of all one hundred subjects from polydat_1.

Fig.19: An example for memorizing as the system "learns"

The point where the recognition rate starts to decrease marks the beginning of overlearning. In practical applications, most adaptive learning algorithms are trained only to the point before overlearning occurs, when the performance on the testing data reaches its peak.

In our testing we had taken that approach and, for each trial, the percentage of correct recognition was taken as the maximum attained for the testing data within forty epochs[37].

We disregarded the recognition rate for the training data because for many systems, including our own, a proper set-up could easily attain a recognition rate of 100%. That is, the recognition rate of the training data bears little importance in practical applications.

[37]An epoch is defined as one complete cycle through all the training data.

§4. RESULTS AND CONCLUSIONS

4.1. Fuzzy-c-means

4.1.1. Searching for the best level of fuzziness (parameter 'm'):

One of the major steps during the one-dimensional clustering was the searching process for the best value of m[38]. For this process, it was necessary to run the FCM algorithm for different m's and for different data by increasing m systematically. This was done for all 669 features and for each polydat_i, by every new m.

Recall that it was decided to consider four groups of features to limit the feature pool for multi-dimensional clustering. Even though the general development - while changing m - was similar for each polydat_i, the individual reaction of these 4 groups within each polydat_i was a little different. For the final decision, we considered all these variances, correct detection rates and also the distributions of the membership values for each m.

In the following, I will mention some of the remarkable observations we have made during this process (see also the following tables and figures representing the results of polydat_3):

As expected, the membership values U_{ik} did approach the 0.5-level[39] by increasing m, i.e. the results became fuzzier. Thus, we had to limit the increasing process to avoid the uncertainty of the results caused by too much "fuzziness" (which means that every person belongs to both clusters with almost the same possibility). However, we could observe a very interesting phenomenon. Even though the membership values came closer to 0.5, and the distances for different persons to this level were around 10^{-x} (with $x > 3$), they were still visually recognizable as deceptive and truthful clusters.

See the following two figures and also the Fig.14 for examples. Notice that the first 50 sessions represent the non-deceptive persons and the other 50 the deceptive ones.

---

[38]See also chapter 3.1.3.3. for the discussion about finding the best m.
[39]See chapter 2.2.3.2. for more details.

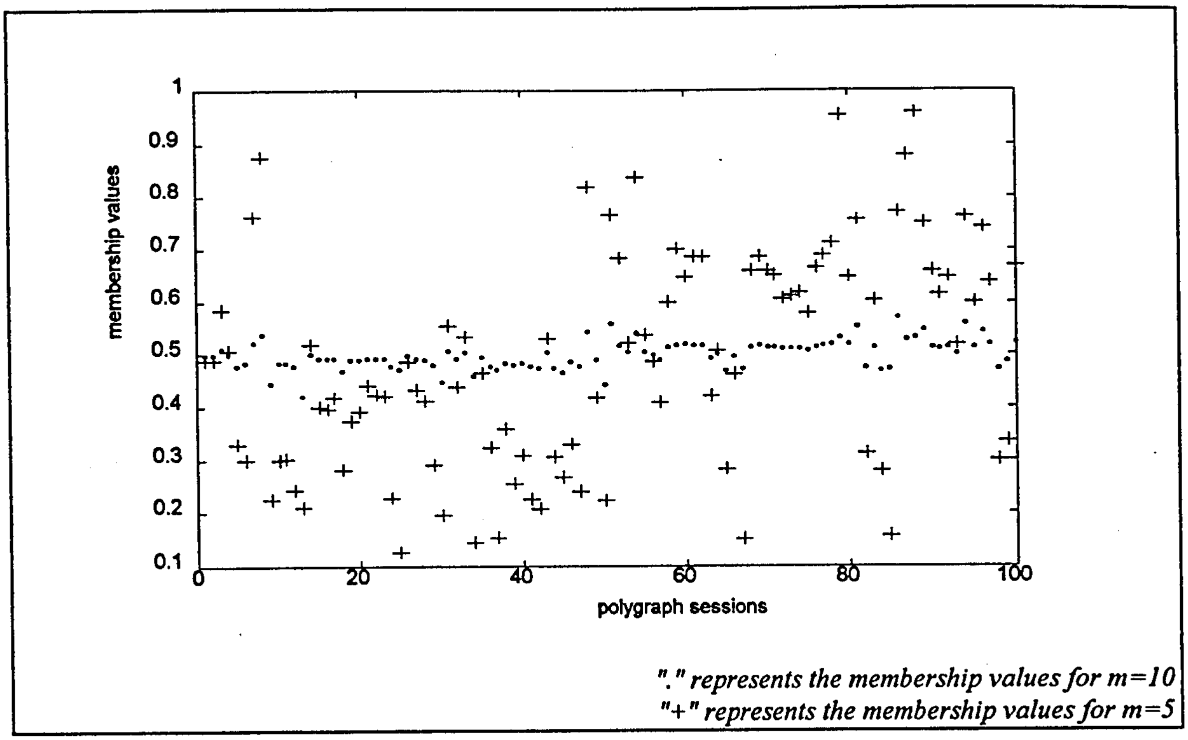


Fig.20: Influence of increasing 'm' for polydat-3 session #1

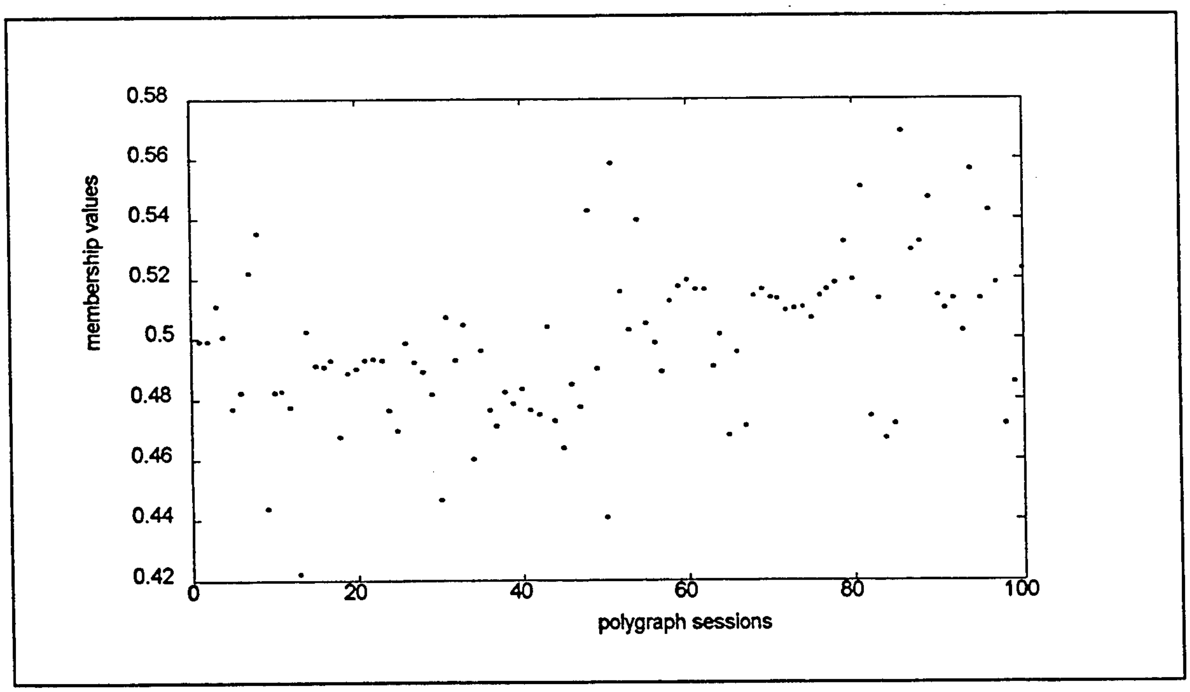


Fig.21: The zoomed-in view of the above figure for m=10

In the following two tables, the influence of changing m (for polydat_3/group #1, as an example) is depicted. As mentioned earlier this group represents those features which give us better than 60% right detection for both deceptive and non-deceptive files by one-dimensional clustering.

As you see in these examples, while increasing the parameter m, new "good" features appear. Some old ones provide even better detection rates and some get worse or even disappear. This progress is not unlimited. As you see, the development from 'm=4' to 'm=5' is smoother than between 'm=2' and 'm=4' regardless of 'm=3' step. By continuing this process above 'm=5', the tendency becomes rather negative.

Those features marked with (*) represent a better detection rate than 75% at least in one of the two clusters. Notice that these features also change during the increasing process of m. By continuing this process above 'm=5', also this tendency becomes rather negative.

After considering the other groups[40] and their development for each polydat_i, 'm=5' appeared to be the best compromise. Notice that there is also an outstanding result for feature number 452 by 'm=5' (see Fig.23). That was the only *inidividual* feature ever by an one-dimensional clustering process with a correct detection rate of 90% for non-deceptive files.

Another interesting aspect is that independent of m, the conglomeration areas where "good" features appear are always the same: For example the half of the "good" features are among the first hundred, but between 200 and 300, there is only one.

In the next tables we will use the following abbreviations:

ft #:	Feature number.
w_dcp:	Wrong detection within the deceptive cluster in percent.
w_non:	Wrong detection within the non-deceptive cluster in percent.
***:**	Features with a better detection rate than 75% at least in one of the two clusters.

'm=...' ***MINUS*** **'m=...':** Represents the difference in detection rates by using different m's.

[40]See Fig.8.

polydat_3

group #1 & m=4				'm=2' *MINUS* 'm=4'	
ft #	w_dcp	w_non		%	%
1.0000	24.0000	18.0000	*	0	-2.0000
3.0000	32.0000	20.0000	*	2.0000	0
4.0000	22.0000	36.0000	*	---------new feature--------	
5.0000	30.0000	32.0000		2.0000	0
12.0000	40.0000	30.0000		---------new feature--------	
15.0000	24.0000	18.0000	*	0	-2.0000
17.0000	32.0000	20.0000	*	2.0000	0
18.0000	22.0000	36.0000	*	---------new feature--------	
19.0000	30.0000	32.0000		4.0000	0
22.0000	24.0000	28.0000	*	0	0
29.0000	36.0000	18.0000	*	0	0
30.0000	24.0000	20.0000	*	-2.0000	4.0000
31.0000	28.0000	32.0000		-2.0000	0
33.0000	36.0000	32.0000		0	0
36.0000	30.0000	16.0000	*	0	0
37.0000	16.0000	26.0000	*	0	4.0000
38.0000	24.0000	28.0000	*	0	0
39.0000	28.0000	26.0000		-6.0000	4.0000
40.0000	32.0000	30.0000		0	0
50.0000	34.0000	34.0000		2.0000	0
52.0000	30.0000	40.0000		---------new feature--------	
68.0000	24.0000	36.0000	*	---------new feature--------	
70.0000	40.0000	40.0000		-2.0000	0
82.0000	32.0000	40.0000		---------new feature--------	
141.0000	34.0000	34.0000		-4.0000	0
155.0000	34.0000	34.0000		-4.0000	0
176.0000	32.0000	36.0000		---------new feature--------	
177.0000	32.0000	32.0000		0	0
197.0000	26.0000	32.0000		0	2.0000
200.0000	34.0000	26.0000		-2.0000	0
202.0000	30.0000	28.0000		0	0
211.0000	26.0000	32.0000		0	0
214.0000	32.0000	26.0000		0	0
216.0000	30.0000	28.0000		0	0
235.0000	30.0000	38.0000		-4.0000	2.0000
395.0000	38.0000	32.0000		---------new feature--------	
449.0000	32.0000	20.0000	*	0	6.0000
450.0000	24.0000	18.0000	*	0	2.0000
451.0000	24.0000	38.0000	*	---------new feature--------	
453.0000	36.0000	36.0000		0	0
458.0000	32.0000	26.0000		6.0000	-2.0000
459.0000	40.0000	18.0000	*	---------new feature--------	
460.0000	26.0000	38.0000		---------new feature--------	
462.0000	28.0000	34.0000		0	0
600.0000	36.0000	40.0000		0	0

for the abbreviations see page 46

Fig.22: Comparison between the results for 'm=2' and 'm=4'

polydat_3

group #1 & m=5				'm=4' *MINUS* 'm=5'	
ft #	w_dcp	w_non		%	%
1.0000	24.0000	18.0000	*	0	0
3.0000	32.0000	20.0000	*	0	0
4.0000	24.0000	36.0000	*	-2.0000	0
5.0000	30.0000	32.0000		0	0
12.0000	40.0000	30.0000		0	0
15.0000	24.0000	18.0000	*	0	0
17.0000	32.0000	20.0000	*	0	0
18.0000	24.0000	34.0000	*	-2.0000	2.0000
19.0000	30.0000	32.0000		0	0
22.0000	24.0000	28.0000	*	0	0
29.0000	36.0000	18.0000	*	0	0
30.0000	24.0000	20.0000	*	0	0
31.0000	28.0000	32.0000		0	0
33.0000	36.0000	32.0000		0	0
36.0000	30.0000	16.0000	*	0	0
37.0000	16.0000	26.0000	*	0	0
38.0000	24.0000	28.0000	*	0	0
39.0000	28.0000	26.0000		0	0
40.0000	32.0000	30.0000		0	0
50.0000	34.0000	34.0000		0	0
52.0000	30.0000	40.0000		0	0
68.0000	26.0000	36.0000		-2.0000	0
70.0000	40.0000	40.0000		0	0
82.0000	32.0000	40.0000		0	0
141.0000	34.0000	34.0000		0	0
155.0000	34.0000	34.0000		0	0
176.0000	32.0000	36.0000		0	0
177.0000	32.0000	32.0000		0	0
197.0000	26.0000	34.0000		0	-2.0000
200.0000	34.0000	26.0000		0	0
211.0000	26.0000	32.0000		0	0
214.0000	34.0000	24.0000	*	-2.0000	2.0000
216.0000	30.0000	28.0000		0	0
235.0000	30.0000	38.0000		0	0
395.0000	36.0000	34.0000		2.0000	-2.0000
449.0000	32.0000	22.0000	*	0	-2.0000
450.0000	24.0000	18.0000	*	0	0
451.0000	26.0000	36.0000		-2.0000	2.0000
452.0000	10.0000	40.0000	*	-----new feature-----	
453.0000	36.0000	36.0000		0	0
458.0000	32.0000	28.0000		0	-2.0000
459.0000	40.0000	20.0000	*	0	-2.0000
460.0000	28.0000	36.0000		-2.0000	2.0000
462.0000	28.0000	34.0000		0	0
600.0000	36.0000	40.0000		0	0

----feature # 202 is missing----

for the abbreviations see page 46

Fig.23: Comparison between the results for 'm=4' and 'm=5'

4.1.2. Searching for the best feature combination:

4.1.2.1. Results of the conventional methods and general observations:

As mentioned in chapter 3.1.3.2.1, we decided for three different strategies to find out the best feature combination that can represent the two sought clusters within the polygraph data.

After a short while of a "trial-and-error" testing with the multi-dimensional clustering algorithm and achieving some experience about how well which features are in a combination with others, I decided to start a systematic searching process beginning with four-tuple combinations. In the followings, I will mention some of the general observations[41] we made;

- not always all of the good one-dimensional features were represented within the best feature combinations,
- good one-dimensional features with the same detection rate did not provide the same results within coequal combinations,
- some poor or average individual features turned out to be the best features in a combination with others,
- by repeating some features in a combination, we obtained a few new good combinations,
- good feature combinations always gave us better results than any of the features individually and
- the quality of the feature tuple does not depend on the order of the features within the tuple.

In the following tables, you see an example for using the random search method for polydat_3 ('m=2' and 'm=5') for four-tuple combinations.

[41]See also chapter 4.3.

feature number = { 1, 4, 3, 9, 22, 29, 30, 36, 37, 39, 450, 457, 458, 460]
condition: if(((nn>=80) & (ww>=80)) | ((nn>=86) | (ww>=86)))

table 1

feature positions				right detection non-ok	dcp-ok
5	1	7	4	86	78
1	7	3	6	88	72
4	8	5	2	86	76
5	6	8	4	86	68
8	3	4	5	86	72
6	8	13	5	86	68
4	1	6	3	88	70
2	3	6	1	86	74
1	8	5	3	86	72
6	12	13	8	86	68
8	1	4	6	86	70
8	7	6	1	86	70
1	8	5	6	86	70
6	3	7	1	88	72
2	6	10	1	86	68
6	10	2	7	86	68
1	3	6	5	88	70
6	7	3	1	88	72
2	6	4	1	86	72
7	5	1	4	86	78
5	8	1	4	86	70
8	5	13	3	86	72
3	8	6	14	88	70
3	7	4	2	86	78
8	7	1	6	86	70
3	1	6	5	88	70
5	4	8	2	86	76

feature positions				right detection non-ok	dcp-ok
6	4	8	5	86	68
2	4	10	6	86	68
8	4	1	5	86	70
10	8	2	1	86	72
7	9	3	1	82	80
8	1	6	14	86	70
5	4	2	8	86	76
1	7	8	6	86	70
1	4	8	10	86	72
2	12	8	1	86	76
1	2	4	8	86	76
8	1	2	4	86	76
7	3	4	2	86	78
4	1	6	8	86	70
3	6	1	4	88	70
8	1	5	10	86	72
1	8	2	4	86	76
8	4	13	1	86	70
1	10	2	6	86	68
1	6	3	5	88	70
1	5	8	3	86	72
3	8	2	6	86	72
1	6	3	14	88	70
5	1	8	2	86	76
1	4	6	10	86	68
2	5	4	8	86	76
2	6	10	1	86	68

...

feature number = {1, 4, 3, <u>8</u>, 9, <u>18</u>, 22, 29, 30, 36, 37, 39, <u>81</u>, 457}
condition: if(((nn>=80) & (ww>=80)) | ((nn>=86) <u>&</u> (ww>=<u>78</u>)))

table 2

feature positions				right detection non-ok	dcp-ok
2	3	9	14	86	78
3	5	2	9	86	78
9	3	2	4	86	78
9	1	4	5	86	78
1	4	13	9	86	78
9	4	3	2	86	78
7	1	4	9	86	78
5	7	9	1	86	78
2	9	3	7	86	78

feature positions				right detection non-ok	dcp-ok
7	1	13	9	86	78
9	3	13	2	86	78
1	9	5	4	86	78
7	3	2	9	86	78
7	9	4	1	86	78
4	2	3	9	86	78
1	7	9	4	86	78
9	1	13	5	86	78

Fig. 24.I: Feature combinations by 'random search' - polydat_3, 'm=2'

feature number = {1, 4, 3, 7, 8, 9, 22, 30, 36, 37, 81, 308, 457, 459}
condition: if (((nn>=80) & (ww>=80)) | ((nn>=86) & (ww>=78)))

table 3

feature positions				right detection non-ok	right detection dcp-ok
8	7	6	1	86	78
7	8	1	5	86	78
3	2	8	6	86	78
3	8	5	2	86	78
1	3	10	8	82	80
3	8	2	6	86	78
3	2	13	8	86	78
2	8	5	3	86	78
1	6	5	8	86	78
5	8	3	2	86	78
1	8	13	5	86	78
6	1	8	7	86	78
2	5	8	3	86	78
5	2	3	8	86	78
3	8	6	2	86	78
3	7	2	8	86	78
2	8	5	3	86	78
7	6	1	8	86	78
3	5	2	8	86	78
8	5	6	1	86	78
7	2	3	8	86	78
8	5	6	1	86	78
7	8	2	3	86	78
7	8	6	1	86	78
8	1	7	6	86	78
1	8	5	6	86	78
1	7	6	8	86	78
5	8	1	6	86	78
6	1	5	8	86	78
7	8	5	1	86	78
8	7	2	3	86	78
8	2	3	7	86	78
6	5	1	8	86	78
1	8	7	6	86	78
6	7	8	1	86	78
1	6	13	8	86	78
6	8	13	1	86	78
8	7	1	6	86	78
5	1	7	8	86	78
2	6	8	3	86	78
3	2	8	7	86	78
1	6	8	5	86	78
2	5	8	3	86	78
8	1	5	7	86	78
2	5	3	8	86	78

feature positions				right detection non-ok	right detection dcp-ok
1	8	10	3	82	80
1	7	8	14	86	78
6	7	1	8	86	78
10	8	1	3	82	80
5	3	2	8	86	78
7	1	6	8	86	78
6	2	8	3	86	78
7	6	8	1	86	78
8	5	3	2	86	78
1	8	6	14	86	78
3	5	8	2	86	78
7	3	8	2	86	78
8	5	2	3	86	78
8	6	7	1	86	78
8	1	5	7	86	78
1	6	13	8	86	78
7	3	8	2	86	78
6	8	1	5	86	78
5	1	8	7	86	78
1	7	13	8	86	78
1	8	5	6	86	78
8	3	2	7	86	78
6	2	8	3	86	78
8	2	3	5	86	78
6	8	2	3	86	78
8	3	6	2	86	78
2	8	3	5	86	78
2	6	3	8	86	78
5	8	1	7	86	78
8	5	13	1	86	78
1	3	8	10	82	80
7	3	2	8	86	78
3	2	5	8	86	78
3	10	1	8	82	80
8	3	1	10	82	80
8	1	5	6	86	78
3	2	13	8	86	78
1	7	8	6	86	78
3	2	5	8	86	78
2	3	8	6	86	78
5	8	13	1	86	78
8	3	13	2	86	78
8	3	5	2	86	78
8	2	3	5	86	78
6	8	2	3	86	78

Fig. 24.I: Continued

feature number = {1, 4, 3, 8, 9, <u>21</u>, 22, 30, <u>35</u>, 36, 81, <u>198</u>, 457, 459}
condition: if(((nn>=80)&(ww>=80)) | ((nn>=86) & (ww>=78)))

table 3

feature positions				right detection non-ok	dcp-ok
1	8	5	4	86	78
7	1	8	14	86	78
7	1	8	5	86	78
4	2	8	3	86	78
3	2	8	5	86	78
8	1	4	7	86	78
3	4	2	8	86	78
8	2	3	7	86	78
5	8	13	1	86	78
1	4	13	8	86	78

feature number = {1, 4, 3, 8, 9, 22, 30, 35, <u>51</u>, <u>111</u>, <u>210</u>, <u>455</u>, 457, 459}
condition: if(((nn>=80)&(ww>=80)) | ((nn>=86) & (ww>=<u>79</u>)))

table 4

feature positions				right detection non-ok	dcp-ok
7	5	10	6	80	80
6	4	7	10	80	80
7	4	10	5	80	80

Fig. 24.I: Continued

feature number = {1, 3, 4, 8, 9, 22, 30, 37, 81, 111, 452, 450, 459, 460}
condition: if(((nn>=80) & (ww>=80)) | ((nn>=86) & (ww>=79)))

table 1

feature positions				right detection non-ok	dcp-ok
1	12	5	9	86	80
5	10	2	8	80	80
6	12	1	9	86	80
1	9	7	5	86	80
10	9	6	7	84	82
7	10	9	6	84	82
2	1	5	8	80	80
10	8	7	6	80	82
7	4	9	1	86	80
1	8	2	4	80	80
1	7	5	9	86	80
8	3	1	10	80	80
5	8	1	2	80	80
8	2	4	10	80	80
5	12	7	3	82	80

feature positions				right detection non-ok	dcp-ok
8	5	1	2	80	80
8	5	2	1	80	80
1	6	2	8	80	80
10	6	2	8	80	80
1	9	7	14	86	80
1	9	8	2	80	80
5	12	9	8	80	80
3	10	8	1	80	80
8	12	1	3	80	80
1	4	8	2	80	80
1	12	13	9	86	80
10	8	2	9	80	80
7	9	6	1	86	80
9	5	7	10	84	82
2	1	4	8	80	80

Fig. 24.II: Feature combinations by 'random search' - polydat_3, 'm=5'

feature number = {1, 4, 8, 9, 22, 30, 32, 37, 67, 81, 452, 450, 459, 457}

condition: if(((nn>=81) & (ww>=81)) | ((nn>=86) & (ww>=79)))

table 2

feature positions				right detection non-ok	dcp-ok
1	6	4	10	86	80
6	4	1	10	86	80
1	12	3	10	86	80
1	12	13	14	86	80
3	6	1	10	86	80
6	10	5	1	86	80
4	6	10	1	86	80
10	3	1	6	86	80
3	12	10	1	86	80
1	12	10	5	86	80
10	12	1	14	86	80

Fig. 24.II: Continued

After running similar simulations for different *m*'s with randomly chosen features from the pool of the aforementioned five[42] groups, I started a sequence of pseudo-exhaustive searches with those features from which we received good results by random search.
For this sequence of simulations the parameter *m* was set equal to 5. We started with four-tuple combinations out of a pool of 14 features (4/14). We then gradually increased the number of the features - within the tuple and the pool - up to 8/22. To run the simulation with this final setting, we needed a computation time of several weeks.

In the following figures, you see an example for one of the best 4-tuple results we obtained for the polydat_3:

<u>4-tuple combination:</u> 81 & 111 & 450 & 452[43].

<u>dimension:</u> polygraph session.
<u>correct detection rate:</u> 84% for non-deceptive and 86% for deceptive files.

<u>dimension:</u> polygraph examination[44] - containing 1 to 4 sessions.
<u>correct detection rate:</u> 89% for non-deceptive and 94% for deceptive files.

<u>dimension:</u> polygraph examinations with more than two sessions.
<u>detection rate:</u> 100%.

---

[42]See Fig. 8 for four of them and page 25 for the additional fifth one.
[43]For information about the exact meaning of these feature *numbers*, see Fig.41.
[44]See "Evaluation strategy" in chapter 3.1.3.4.

Uik	defuzzification per session	test	
0.2727	0		
0.4680	0		
0.4404	0		
		0	
0.5774	1.0000		
0.3208	0		
0.4075	0		
		0	
0.6157	1.0000		
0.5416	1.0000		
		1	*misclustered*
0.4095	0		
0.4480	0		
0.4862	0		
		0	
0.4722	0		
0.4755	0		
0.5046	1.0000		
		0	
0.4387	0		
0.4459	0		
0.4346	0		
		0	
0.4005	0		
		0	
0.4351	0		
0.4251	0		
0.3723	0		
		0	
0.4505	0		
0.4414	0		
0.3218	0		
		0	

0.4428	0		
0.4474	0		
0.5997	1.0000		
		0	
0.3764	0		
0.3709	0		
0.3383	0		
		0	
0.4668	0		
0.4843	0		
0.4515	0		
		0	
0.3964	0		
0.5232	1.0000		
0.4085	0		
		0	
0.3915	0		
0.4425	0		
0.3860	0		
		0	
0.4200	0		
0.4443	0		
0.4315	0		
		0	
0.4974	0		
0.3980	0		
0.3964	0		
		0	
0.5863	1.0000		
		1	*misclustered*
0.3786	0		
0.5783	1.0000		
0.4377	0		
0.3527	0		
		0	

non-deceptive files
polydat_3
m=5

Fig.25: Defuzzified results for [81-111-450-452] feature combination

Uik	defuzzification per session	test
0.6374	1.0000	
0.5389	1.0000	
0.5094	1.0000	
		1
0.5696	1.0000	
0.4185	0	
0.5057	1.0000	
		1
0.5508	1.0000	
0.5237	1.0000	
		1
0.5533	1.0000	
0.5878	1.0000	
0.5941	1.0000	
		1
0.4533	0	
0.5383	1.0000	
0.5316	1.0000	
		1
0.5452	1.0000	
0.5266	1.0000	
0.3128	0	
		1
0.5068	1.0000	
0.5735	1.0000	
0.6276	1.0000	
		1
0.5504	1.0000	
0.5706	1.0000	
0.5542	1.0000	
		1
0.5555	1.0000	
0.5692	1.0000	
0.5650	1.0000	
		1
0.4418	0	
0.6468	1.0000	
0.5009	1.0000	
		1
0.5593	1.0000	
0.5596	1.0000	
0.4109	0	
		1
0.6002	1.0000	
0.5550	1.0000	
0.5148	1.0000	
		1
0.5964	1.0000	
0.6112	1.0000	
0.6224	1.0000	
		1
0.7130	1.0000	
0.5834	1.0000	
0.5844	1.0000	
		1
0.5472	1.0000	
0.5758	1.0000	
0.5924	1.0000	
		1
0.5879	1.0000	
0.6284	1.0000	
0.6078	1.0000	
		1
0.3902	0	
0.5399	1.0000	
0.4636	0	
		0 *misclustered*

deceptive files
polydat_3
m=5

Fig.25: Continued

4.1.2.2. Results of the genetic method:

Simultaneously to the aforementioned sequence of searches, I started with a compromise between the random and the pseudo-exhaustive search method; i.e. the genetic alternative. I decided to use this method in two different ways:

1. In order to increase the number of potentially good features in the pool, I initialized the genetic code with up to 50 features from which (in different simulations) 4-, 6-, 8-tuple combinations were made.

2. In order to accelerate the search, but process the data more exhaustively, I decided to use the genetic code only for the best features from random and pseudo-exhaustive simulations and narrow the feature pool to these 30 selected features. In this simulation, 15-tuple combinations were made.

Recall that having 30 or 50 features in the pool makes a big computation difference. For example, choosing exhaustively 8-tuples out of 50 or 30 features makes a difference of following number of computations:

$$\binom{50}{8}-\binom{30}{8}=\frac{50!}{8!(50-8)!}-\frac{30!}{8!(30-8)!}\approx 5\cdot 10^{8}$$

In the first part of the genetic search - as expected - we had similar problems as scientists have with the theory of evolution as the cause of our being[45]. The only way we could get the following good results was the continuous manipulating of the evolution process - by changing parameters (like mutation rate), features (=genes) and feature numbers (=population size and also number of genes in one chromosome), or by starting again if the simulation began with a very low detection rate (=average fitness). In spite of these manipulations the first version of the genetic search took a simulation time of over two months of continuous computation. Without the constant *controlling* process over this genetic system the *evolution* (by chance as it is its nature) could have hardly provided any appropriate improvement[46]. As a result we obtained 12 (see Fig.26) 8-tuples combination

[45]Further discussion about "evolution vs. creation" would break up the limitations of this project; For interested readers I recommend the following references: [Morris1987] [Johnson1991].

[46]For example, one of the *uncontrolled* simulation for polydat_1 was stopped after 561 generations providing no particular results.

with an average of 85% correct detection rate for polydat_3 similar to the results of the 4-tuple combination mentioned in chapter 4.1.2.1. We also obtained 3 outstanding (86% correct detection rate) individuals within three different generations (population size of 200 to 300, total number of generation 1000, polydat_3).

feature numbers of the best 8-tuple combinations	**correct detection rate ndcp**	**dcp**
8 , 30 , 81 , 81 , 111 , 363 , 458 , 482	84	86
9 , 37 , 81 , 111 , 111 , 449 , 458 , 460	84	86
9 , 37 , 111 , 111 , 449 , 457 , 457 , 482	84	86
9 , 37 , 111 , 111 , 358 , 449 , 457 , 458	84	86
9 , 37 , 111 , 111 , 235 , 449 , 457 , 460	84	86
37 , 79 , 111 , 111 , 197 , 358 , 449 , 457	84	86
37 , 111 , 111 , 197 , 449 , 457 , 460 , 460	84	86
37 , 111 , 111 , 111 , 235 , 358 , 457 , 458	84	86
37 , 111 , 111 , 235 , 235 , 449 , 453 , 457	84	86
37 , 111 , 111 , 197 , 358 , 361 , 458 , 460	84	86
37 , 81 , 111 , 235 , 235 , 363 , 450 , 453	86	84
37 , 81 , 111 , 235 , 235 , 359 , 450 , 453	86	84
37 , 79 , 111 , 111 , 197 , 235 , 449 , 457	86	86
37 , 111 , 111 , 235 , 235 , 453 , 457 , 460	86	86
37 , 111 , 111 , 197 , 235 , 452 , 457 , 460	86	86

ndcp: non-deceptive files
dcp: deceptive files
data: polydat_3

Fig.26: Results of the first version of the genetic search

Concerning the *defuzzified* results, all the combinations with 85% correct detection rate show similar structure as depicted in Fig.25. The three best 8-tuple combinations (86% correct detection rate) cluster the data exactly in the same groups as shown in the following figure.

Uik	defuzzification per session	test	
0.4143	0		
0.4780	0		
0.4583	0		
		0	
0.5269	1.0000		
0.4035	0		
0.4035	0		
		0	
0.5601	1.0000		
0.5412	1.0000		
		1	*misclustered*
0.4391	0		
0.4465	0		
0.4833	0		
		0	
0.4669	0		
0.4679	0		
0.5058	1.0000		
		0	
0.4401	0		
0.4392	0		
0.4481	0		
		0	
0.4114	0		
		0	
0.4405	0		
0.4212	0		
0.4664	0		
		0	
0.4523	0		
0.4488	0		
0.3645	0		
		0	
0.4565	0		
0.4853	0		
0.5849	1.0000		
		0	
0.4441	0		
0.4471	0		
0.3506	0		
		0	
0.4983	0		
0.4872	0		
0.4938	0		
		0	
0.4008	0		
0.4962	0		
0.4058	0		
		0	
0.4268	0		
0.4740	0		
0.4050	0		
		0	
0.4475	0		
0.4517	0		
0.4440	0		
		0	
0.5692	1.0000		
0.4432	0		
0.4118	0		
		0	
0.4289	0		
		0	*compare to Fig.25*
0.4271	0		
0.5548	1.0000		
0.4696	0		
0.4135	0		
		0	

non-deceptive files
polydat_3
m=5

Fig.27: Defuzzified results for [37-111-111-197-235-452-457-460] feature combination

Uik	defuzzification per session	test	
0.5842	1.0000		
0.5511	1.0000		
0.5197	1.0000		
		1	
0.5665	1.0000		
0.5483	1.0000		
0.6586	1.0000		
		1	
0.5227	1.0000		
0.5169	1.0000		
		1	
0.5519	1.0000		
0.5727	1.0000		
0.5747	1.0000		
		1	
0.5411	1.0000		
0.5224	1.0000		
0.6020	1.0000		
		1	
0.4308	0		
0.4916	0		
0.4801	0		
		0	*misclustered*
0.5044	1.0000		
0.5686	1.0000		
0.5830	1.0000		
		1	
0.5488	1.0000		
0.5460	1.0000		
0.5413	1.0000		
		1	

0.5446	1.0000		
0.5495	1.0000		
0.5615	1.0000		
		1	
0.5345	1.0000		
0.5666	1.0000		
0.5370	1.0000		
		1	
0.5539	1.0000		
0.5565	1.0000		
0.4388	0		
		1	
0.5817	1.0000		
0.5042	1.0000		
0.4946	0		
		1	
0.5706	1.0000		
0.5990	1.0000		
0.6133	1.0000		
		1	
0.6386	1.0000		
0.5674	1.0000		
0.5576	1.0000		
		1	
0.5457	1.0000		
0.5646	1.0000		
0.5482	1.0000		
		1	
0.5096	1.0000		
0.5954	1.0000		
0.6347	1.0000		
		1	
0.4532	0		
0.4323	0		
0.5457	1.0000		
		1	*compare to Fig.25*

deceptive files
polydat_3
m=5

Fig.27: Continued

The followings are the clustering results of the best 8-tuple combinations for polydat_3:

dimension:	polygraph session[47].
correct detection rate:	86% for both non-deceptive and deceptive files.
dimension:	polygraph examination - containing 1 to 4 sessions.
correct detection rate:	94% for both non-deceptive and deceptive files.
dimension:	polygraph examinations with more than two sessions
detection rate:	97%.

In the second part of the genetic search as we fed the evolution process with the best features, we obtained after about 3 weeks of continuous simulation the following results:

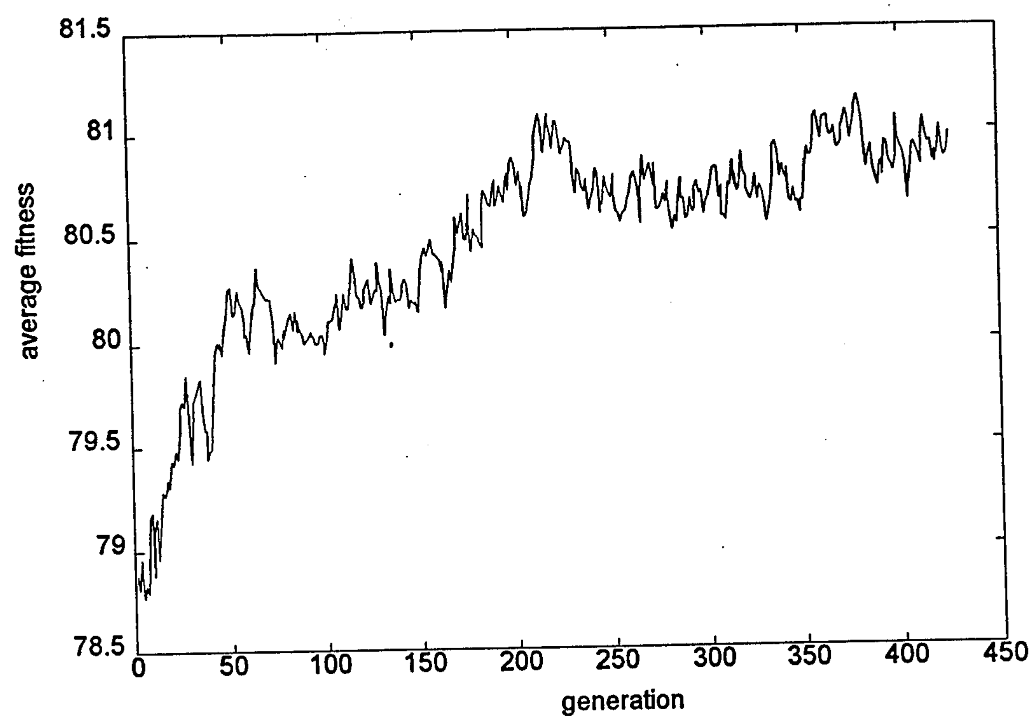
twelve 15-tuple combinations: (the features in each tuple are ordered vertically)

	37	11	8	8	37	30	11	30	11	11	11
	111	11	11	37	81	32	30	32	30	30	30
	111	36	37	50	81	32	32	39	32	32	32
	197	36	111	79	81	32	39	81	39	39	39
	358	37	111	111	81	36	81	81	81	79	81
	358	37	197	111	197	37	81	81	81	81	81
	361	67	235	235	235	39	81	111	81	81	81
	361	81	358	235	358	50	111	197	111	81	111
	449	197	359	358	359	67	197	235	197	111	197
	457	235	359	452	450	79	235	235	235	197	235
	458	457	363	453	450	359	235	358	235	235	235
	458	458	363	478	453	449	358	358	358	235	358
	478	482	452	478	458	449	359	450	358	358	359
	478	482	478	478	478	478	450	478	450	359	450
	482	482	482	482	478	478	482	482	482	450	478
correct detection rates (in %):											
	84	84	84	84	84	84	84	84	84	84	84 **:non-deceptive files**
	86	86	86	86	86	86	86	86	86	86	86 **:deceptive files**

polydat_3, m=5

Fig.28: Results of the second version of the genetic search

[47] See "Evaluation strategy" in chapter 3.1.3.4.



polydat_3, m=5.
15-tuple combinations out of a pool of 30 features[48].

Fig.29: Average fitness of each generation provided by the second version of the genetic search

As you see in this figure, the average fitness (from all the chromosomes within a generation) increases over the period of time. It then approaches a local asymptote which represents a local error minimum. By increasing the mutation rate after the 150th generation, we could avoid being stuck in that local minimum for further development. This higher mutation rate helped the evolution process getting a 1% better average fitness per generation for the rest of the simulation.

Our hope for this simulation was to get *outstanding* chromosomes with a very high fitness simultaneously to the increasing process of the *average* fitness per generation. However, the outstanding chromosomes appeared unsystematically in different generations and not at the end. In fact, most of them[49] belong to the first part of this evolution.

---

[48]See the begining of this chapter for more details.
[49]See Fig.28 for the best feature combinations.

4.1.2.3. Final results of FCM- A comparison between all three polydat_i's:

All the aforementioned results belong to the data set polydat_3, and all the three methods, (1) previous researches using the fuzzy K-nearest neighbor (KNN) classifier, (2) the LMS fuzzy adaptive filter and also (3) the fuzzy-c-means algorithm show that the data structure within the polydat_3 is better to cluster or classify than the other two sets.

As it is the nature of a *clustering* versus a *classifying* method, I did not set the highest priority on finding the *same* best features for all three polydat_i's, but for each of them individually. After finding those best combinations, I then compared the results and tested the consistency of the features (see Fig. 33, 34, 35).

Using either *sessions* or *examinations*[50] as the counting dimension the best results for each polydat_i individually are shown in the following figures.

data	**average correct detection rate**
polydat_1	81%
polydat_2	79%
polydat_3	86%

Fig.30: Clustering results using individual features (using *sessions* as the counting dimension)

data	**average correct detection rate**
polydat_1	91%
polydat_2	82%
polydat_3	94%

Fig.31: Clustering results using individual features (using *examinations* as the counting dimension)

[50] See "Evaluation strategy" in chapter 3.1.3.4.

data	average correct detection rate
polydat_1	93%
polydat_2	87%
polydat_3	97%

Fig.32: Clustering results using individual features (counting only those *examinations* with more than two sessions)

In the following figures, a comparison between the three polydat_i's were made using the best feature combination for one of the polydat_i's at a time and testing it for the other two ones. As you will see, the best result[51] - while taking the *same* features for each polydat_i - is 79.7% for the feature combination[52] [9, 30, 81, 197, 478, 111], and in average 79.3%.

feature tuple	polydat_i i=3	i=2	i=1
37, 79, 111, 111, 197, 235, 449, 457	86%	77%	75%
37, 111, 111, 197, 235, 452, 457, 460	86%	77%	75%
37, 111, 111, 235, 235, 453, 457, 460	86%	77%	74%
30, 81, 81, 111, 197, 458	85%	79%	73%
9, 30, 81, 111, 197, 458	85%	79%	73%
8, 37, 50, 79, 111, 111, 235, 235, ... 358, 452, 453, 478, 478, 478, 482	85%	76%	76%

Fig.33: Comparison #1 (dimension: sessions) (taking some of the best *polydat_3* feature tuples and testing it for the others)

For the exact *labels* of this feature *numbers* see appendix, Fig.42.

[51] With polygraph *sessions* as the counting dimension.
[52] See Fig.35, "Comparison #3".

feature tuple	polydat_i i=1	 i=2	 i=3
9, 30, 30, 39, 235, 450	80%	75%	81%
30, 30, 39, 50, 235, 450	80%	75%	81%
30, 30, 39, 81, 235, 450	80%	75%	81%
30, 30, 39, 197, 235, 450	81%	74%	82%
30, 30, 39, 235, 363, 450	81%	75%	81%
30, 30, 39, 235, 358, 450	80%	76%	81%
30, 30, 39, 235, 450, 458	80%	75%	81%
30, 30, 39, 235, 482, 450	80%	75%	81%
30, 30, 39, 235, 361, 450	80%	75%	81%
30, 30, 39, 235, 359, 450	80%	75%	81%
30, 30, 39, 235, 450, 457	80%	75%	81%
30, 39, 235, 363, 450, 482	80%	72%	83%
30, 39, 235, 363, 450, 478	80%	71%	83%

Fig.34: Comparison #2 (dimension: sessions)
(taking some of the best *polydat_1* feature tuples and testing it for the others)

feature tuple	polydat_i i=2	 i=1	 i=3
9, 30, 81, 197, 478, 111	79%	75%	85%
9, 30, 50, 81, 197, 111	79%	74%	85%
9, 30, 81, 358, 197, 111	79%	74%	85%
9, 30, 81, 359, 197, 111	79%	74%	85%
9, 30, 81, 197, 457, 111	79%	74%	85%
30, 81, 105, 111, 197, 358	79%	74%	84%
30, 81, 105, 111, 197, 359	79%	74%	84%
30, 81, 105, 111, 197, 457	79%	74%	85%
30, 81, 105, 111, 197, 459	79%	74%	84%
30, 81, 111, 197, 358, 359	79%	74%	85%
30, 81, 111, 197, 358, 456	79%	74%	85%
30, 81, 111, 197, 358, 457	79%	74%	85%
30, 81, 111, 197, 358, 459	79%	74%	85%
30, 81, 111, 197, 359, 456	79%	74%	85%
30, 81, 111, 197, 359, 457	79%	74%	85%
30, 81, 111, 197, 359, 459	79%	74%	85%
30, 81, 111, 197, 456, 457	79%	73%	85%
30, 81, 111, 197, 456, 459	79%	74%	85%
30, 81, 111, 197, 457, 459	79%	74%	85%
30, 105, 111, 197, 359, 459	79%	74%	84%
30, 105, 111, 197, 456, 459	79%	74%	84%
30, 105, 111, 197, 457, 459	79%	74%	85%
30, 105, 111, 197, 456, 457	78%	74%	85%
30, 111, 197, 358, 359, 459	78%	74%	85%
30, 111, 197, 358, 456, 459	78%	74%	85%
30, 111, 197, 358, 457, 459	78%	74%	85%
30, 111, 197, 456, 457, 459	78%	74%	85%

Fig.35: Comparison #3 (dimension: sessions)

(taking some of the best *polydat_2* feature tuples and testing it for the others)

4.2. LMS fuzzy adaptive filter

The first test we did, was to find the performance of the filter before any training. That is, we used the classifier as a conventional fuzzy logic system designed solely based on the four linguistic rules mentioned above. The results are listed in the following table:

	correct detection rate in		
polydat_i	**non-deceptive files**	**deceptive files**	**average**
i=1	70%	72%	71%
i=2	70%	76%	73%
i=3	70%	88%	79%

Fig.36: Results based solely on 4 aforementioned linguistic rules without any training

Note that the percentage of correct recognition for non-deceptive subjects are the same for polydat_1, polydat_2, and polydat_3, because they are all the same data[53]. Also note that the results are best for polydat_3, as it was for KNN and FCM. This may be partially due to polydat_3's good performance in general, independent of the classifying schemes. We believe that it may also be a result of us setting up the linguistic rules by having observed polydat_3.

However, the outcomes for polydat_1 and polydat_2 are good enough such that one can be sure the linguistic rules are sufficiently general even for data that we did not examine.

As mentioned in chapter 3.2.3, we then tested the fuzzy LMS algorithm trained with twenty training data (ten deceptive and ten non-deceptive) and again with seventy training data (thirty-five deceptive and thirty-five non-deceptive) for the three sets of data, for a total of six tests. Twenty trials were performed for each test, and the system was initialized with the linguistic rules before each trial. The training data were randomly chosen for each trial, and the rest of the available data in each set were for testing.

[53]See polygraph files on chapter 6.2.

We computed the percentage of correct recognition of testing data for each trial, averaging the performance for deceptive and non-deceptive subjects. The recognition rate of those twenty trials are averaged, rounded to two digits, and reported in the following table. The sample standard deviations are also shown.

	correct detection rate	
polydat_i	**version #1**	**version #2**
i=1	75% (6%)	73% (2%)
i=2	74% (7%)	73% (3%)
i=3	78% (6%)	79% (2%)

version #1: 70 training & 30 testing sessions
version #2: 20 training & 80 testing sessions
(standard deviation in parentheses)

Fig.37: Average percentage of correct detection rate for twenty trials of each test

As may be expected, the recognition rate improves in general when training data is used, as compared to the results of the untrained system. Also, the recognition rate is typically higher when the system is trained with more data. The difference, however, is not dramatic. The use of training data offers small incremental improvements. The one exception would be for data set polydat_3. Here more training data seems to lower the performance. The effect is probably due to the fact that the initialization of the reasoning rules were based on our examination of polydat_3, which covered all 100 data. Yet the training algorithm was to learn only a subset of that, so it was handicapped compared to human reasoning.

Human reasoning may also be better in this case because the training algorithm only attempts to optimize the system in the least mean square sense, slightly different than our ultimate goal of maximizing recognition rate. At any rate, when the standard deviation is taken into account, the difference in recognition rate becomes insignificant.

Another noticeable difference between the results using different amounts of training samples is the value of the sample standard deviation. A large number of testing data leads

to a small standard deviation. Conversely, a small amount of testing data leads to a large standard deviation. This confirms what we intuitively know; the average percentage of correct recognition is more accurate when a large amount of testing data is available.

The above observations illustrate a practical issue in using many adaptive and learning algorithms, that of partitioning a limited amount of data into training and testing sets. For most algorithms, too much data in training and little in testing leaves little assurance about the performance of the system. On the other hand, too much data in testing and little in training assures mediocre performance from the system.

More data for both training and testing would help, but many times that may not be available. Fuzzy logic systems mitigate this problem by exploiting linguistic information. Unlike neural networks and many statistical techniques, which are *completely* dependent on numerical data, this fuzzy LMS algorithm uses numerical data mainly to optimize a good fuzzy system. The above results show that, given good initialization of the reasoning rules, the system can perform well even with little or no training data. This robustness is one of the many advantages of fuzzy logic.

4.3. Other observations:

During this project, aside from the results and conclusions we were looking for, we also obtained several side results. In this passage, I will mention some of the interesting observations we made.

1. As mentioned before, the fuzzy-c-means (FCM) algorithm is initialized by random chosen membership values which will be modified and optimized during the iterative process. Thus, FCM algorithm is *almost* independent of the initial membership values. During our testing process, we noticed that the FCM algorithm is not ***absolutely*** independent of the initial values. Thus, it is possible that

- the algorithm may run into different local minima or
- because of its unsupervised nature, the algorithm may switch the clusters, i.e. if - depending on our interpretation - the first cluster represents the non-deceptive and the second one the deceptive files, it might be the opposite while using other initial random values.

To avoid any misinterpretations, I decided to create two sets of random membership values (for c=2 and c=3) and save them as fixed initialization values for any further simulations. In the following figure, '+' represents the non-deceptive, '*' the decptive files;

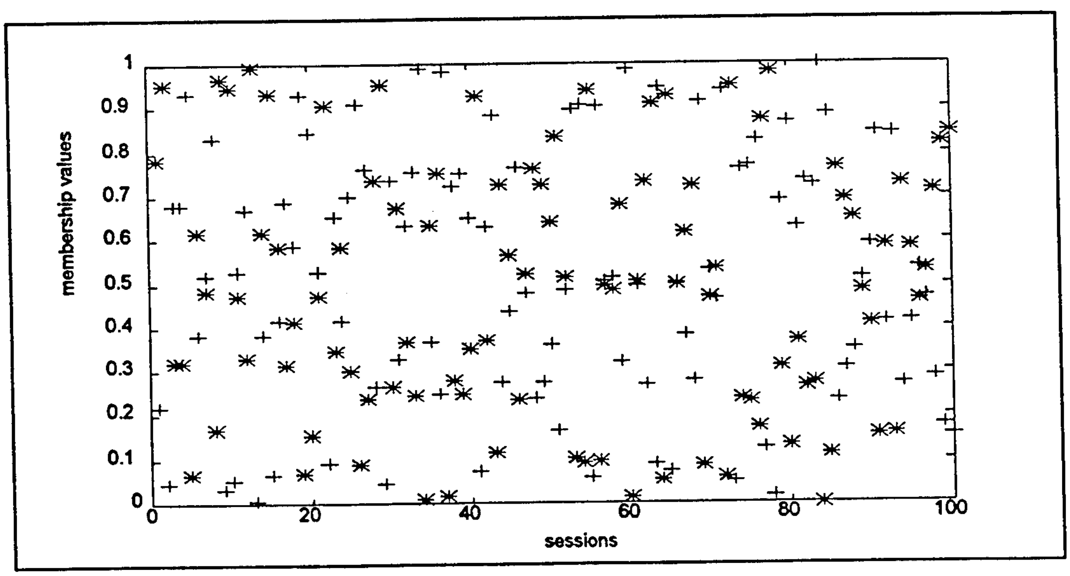


Fig.38: Fixed initial random membership values for c=2

2. "Outlier effect":

In the real world of using an automated polygraph system as suggested in this project, we have to keep in mind the existence of the outlier effect. This occurs, for instance, when a non-deceptive person (= membership value between zero and 0.5) becomes misclustered in a deceptive data space with a very high membership value close to one. In other words, if a normal non-deceptive person gets labeled as *very* deceptive, or vice-versa.

We noticed this phenomenon in both clustering and classifying algorithms[54]. We also noticed that by making the system "fuzzier" - e.g. higher m or/and c for FCM - as expected, the outlier effect can be reduced, but not eliminated though.

3. "Performance limitations":

There seem to be a limit in recognition rate using the features available by both fuzzy algorithms used in this project and also by fuzzy k-nearest neighbor algorithm used in previous works [Layeghi1993,1] [Dastmalchi1993] for all the available polydat_i's. There may also be psychophysiological limitation on the recognition rate. However, polydat_3 provided, independent of all the three algorithms, the best results compared to the other two polydat_i's.

[54]See also *"Epilogue"*.

4.3. A COMPARISON

BETWEEN THE THREE FUZZY ALGORTHMS USED IN THIS AND THE PREVIOUS PROJECT

(FUZZY-C-MEANS, LMS FUZZY ADAPTIVE FILTER AND FUZZY K-NEAREST NEIGHBOR)

The fuzzy LMS system is unique in its application of linguistic knowledge. As mentioned earlier, the use of linguistic knowledge ensures the robustness of the fuzzy system. The use of linguistic information also ameliorates the problem of not having enough reliable numerical data. Unlike classification schemes such as the K-Nearest Neighbor, the fuzzy LMS algorithm is not entirely dependent on numerical data.

When applied to pattern recognition, fuzzy *logic* systems can be set up to perform like KNN systems. In KNN systems, numerical data of known class patterns are set up to estimate the probability density distribution of the classes. The probabilities of new data points belonging to the different classes are then computed based on such distribution. Data points around known class samples are then classified into the same class with a higher probability. The fuzzy-KNN algorithm modifies the classical KNN algorithm by taking into account the distance between the data point and the known class patterns when estimating the probability. Conceptually this is similar to setting up clusters around all known class samples and calculating the degree of belonging of new data points in the different types of clusters. Other than the exact mathematical equations, that description fits a fuzzy adaptive system where each rule corresponds to a known class pattern and the size of the clusters is the same for all rules.

However, fuzzy adaptive systems give up some of the nice theoretical understandings of the KNN systems but gain some practical advantages. The number of rules required are usually much smaller than the number of known samples. Fuzzy *logic* can usually exploit that to reduce system complexity.

Furthermore, the system complexity for a fuzzy adaptive system stays the same even as new information are available. This is partly a result of the way this algorithm adapt continuously; new information are learned as old ones are forgotten. The fuzzy LMS learning technique is like backpropagation, a popular neural network training technique. However, the fuzzy LMS learning algorithm requires few epochs for training. In all our

trials the maximum recognition rates for testing data peaked in less than thirty epochs. About 95% of them peaked in less than twenty epochs[55]. This is a few orders of magnitude less than most applications of backpropagation. In many cases the peaks occurred before any training; that is, the system uses only linguistic rules. Here the use of expert knowledge speeds up the training of the system.

The fuzzy-c-means algorithm, unlike fuzzy LMS, is an unsupervised clustering algorithm. Given a set of data, FCM looks for a (usually) predetermined number of clusters within the data points. It does not use any knowledge about the correct, or desired classification of any of the elements. The algorithm only minimizes an objective function, which is the sum of a function of the data points' membership values and the distances between the data points and the clusters' centers.

FCM operates like a black box; given some data, the algorithm automatically computes the results[56]. This presents the advantage that different sets of data using different features can be tested in a routine manner. FCM also presents a way to normalize the different dimensions of the data, just like the use of sigma in the fuzzy LMS algorithm. However, unlike fuzzy LMS, FCM does not present a method to find the optimal way for such normalization.

The fuzzy LMS algorithm, however, does pose some potential problems of its own. The use of expert knowledge, while a benefit in some senses, may not be always straightforward. For example, in our project we did not have any specific knowledge about the polygraphy itself. Whatever we learned, we learned by looking at numerical data. As we tried to find more complicated patterns, patterns involving three, four, or more features, the analysis became more difficult. Naturally one wishes to automate this process. If we do not rely on some learning procedures, however, rules cannot be automatically found for the fuzzy system. Much research also needs to be done to understand the fuzzy LMS algorithm's learning dynamics. While the same method, gradient descent, is used on both backpropagation and the fuzzy LMS algorithm, the general shapes of the error surface between the two are different. In backpropagation, all the parameters have the same range and lie in an uniform neural network structure. In the fuzzy LMS algorithm, the parameters can have different ranges and lie a fuzzy logic

[55]However, we ran every trial to forty epochs to ensure that there is no "false" peak.
[56]Our job is basically to adjust the parameters.

structure that is not completely uniform. The effects of such differences on the shape of the error surface and the learning dynamic are unknown.

In the following, I will mention again some of the results we obtained by using different fuzzy clustering or classifying algorithms. Recall that also the searching strategies to find the best features -and feature combinations- were different for each of the aforementioned algorithms[57].

	polydat_i		
	i=1	**i=2**	**i=3**
fuzzy-c-means[58]	91%	82%	94%
fuzzy-c-means[59]	93%	87%	97%
fuzzy K-nearest-neighbor	86%	80%	91%
LMS fuzzy adaptive filter	81%	83%	83%
fuzzy-c-means[60]	81%	79%	86%

The results are rounded.

Fig.39: Comparison between different fuzzy algorithms used for polygraph classification in this and in the previous research

The results of our fuzzy LMS system, while impressive for such a simple set-up, are not comparable to the results of the same project using other systems. We believe that the recognition rate will increase for few percentage points by using the suggestions in chapter 5.1.

[57]See the following chapters 3.1.3.1, 3.1.3.2.1 - 4 for the searching strategies used for the FCM, chapter 3.2.1 for the visual inspection used for the LMS system, and chapter III.3.3. in [Layeghi1993,1] for the methods used for the KNN.

[58]FCM using *examinations* as the counting dimension (see chapter 4.1.2.3. and Fig.31).

[59]The same as above but counting those examinations with more than 2 sessions (see Fig.32).

[60]Since we took 35 out of 50 available non-deceptive *sessions* for training the LMS filter, it would be meaningless to evaluate this algorithm by *examinations* as the counting dimension. Yet, in order to make it comparable to the other algorithms, the results of the FCM with *sessions* as the counting dimension are also shown.

§5. FUTURE STEPS AND SUGGESTIONS

5.1. The algorithms:

As mentioned earlier in chapter 2.2.3. about the fuzzy-c-means algorithm, the performance of this clustering model is influenced by the choice of various parameters. In this project, I tried to find the optimum values of the majority of them. However, there are several other points which should be studied more comprehensively: They are

- the initial cluster centers,
- the order in which the samples are taken as input,
- the choice of distance measure,
- the termination criteria and
- the geometrical properties of the data.

Most imprtantly, more information about the geometrical arrangement of the data points and the appropriate choice of the norm could help us improve the clustering algorithm. There are several suggestions in [Bezdek1981] [Bezdek1992] [IIScorp1993] for a better understanding of the algorithm's dynamics and for making systematic decisions concerning different types of distance norms and elliptical cluster shapes.

For future studies, I highly recommend a deeper investigation of our clustering algorithm by setting c=3 and trying defuzzification thresholds other than 0.5.

In this project, we decided to systematically test the FCM algorithm with different values of m to find its optimum. For additional (and more theoretical) investigations, I suggest [Choe1992] as an introductory step. It may be also helpful to use different values of m for different *sessions* simultaneously, while looking for the most realistic clusters within the entire session *space*.

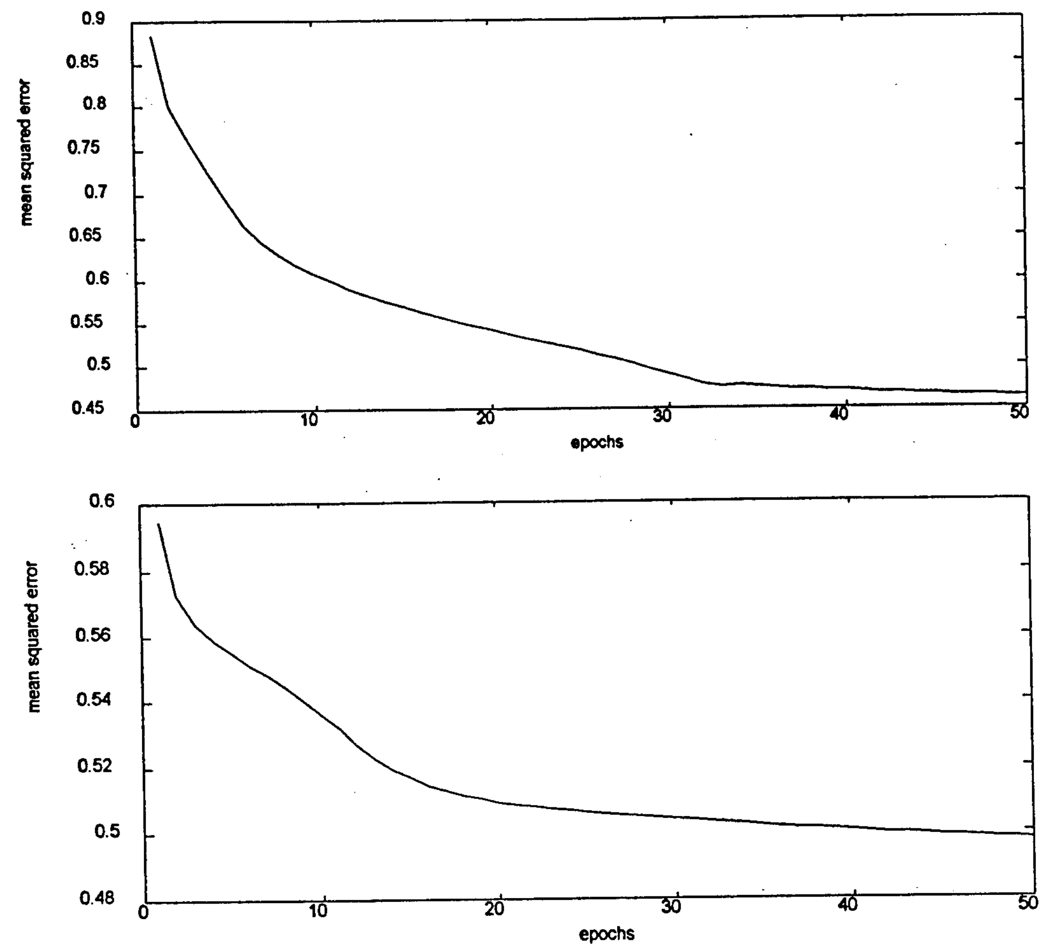
An exciting additional investigation would be a new polydat made up of the best clustered sessions of our three polydat_i's as a reference for any further clustering process. By doing this we could give the algorithm a better chance to cluster correctly even the critical sessions.

Concerning the LMS adaptive algorithm, one may investigate the effect of changing the learning factor; throughout our experiment it remained at 0.005. Upon observing the quickness of learning in our testing, we believe the learning factor can be decreased in the future.

We also believe that there should not be just one but at least three different learning factors: one for the σ's, one for the θ's, and one for the x_i's; because these three types of parameters lie in a very irregular parameter space, unlike that of backpropagation where all parameters lie in a more or less uniform parameter space.

For illustration, the three types of parameters comapred to one another have very different numerical ranges. Conceptually speaking, a parameter with a large range of movement should generally have a larger learning factor than one with a smaller range of movement. However, the gradient and the general shape of the error surface would also affect the value of the learning factors. It is possible that with a constant learning factor, a factor that is too large for one type of parameter - one that causes oscillation for that parameter - may be too small for another type of parameter and effects little change. That is, some parameters become more willing to adapt while others hesitate to change.

Setting up separate learning factors for the different types of parameters should eliminate this problem. However, choosing a learning factor is still a complex trial-and-error task, and having more learning factors to deal with requires more sophisticated understanding of the learning dynamics we possess. Plots of the mean squared error of two sets of randomly chosen training data suggest that there are noticeable points where the rate of decrease dramatically changes (see the following figure).



Two cases of the mean squared error of training data as the system "learns."

Fig.40: The influence of the learning factor

More rules and features should be added to improve this LMS system. As the complexity of the system grows, however, the design will depend more on the learning algorithm than on heuristic knowledge. This requires much more understanding of the learning dynamics. Preliminary testing with three features and eight rules shows little improvement in recognition rate. Obviously many additional studies need to be done in this case.

As mentioned in chapter "Setting Linguistic Rules", for future investigations one may also experiment with different decision thresholds for determining deception and nondeception. However, the benefit, if any, of this is not clear. One may also experiment with mapping the fuzzy output to a confidence value in addition to just a deception/nondeception decision. This may be more helpful in practical situations. One should also test the

algorithm with random initializations; that is, without using any expert knowledge. It would be interesting to compare the training time, performance, and robustness of that system to the present one.

Fuzzy *logic* systems promote rapid development of robust, simple, and reliable systems. Our project validated that point. Some of the main problems with designing *traditional* fuzzy logic systems, however, are their dependence on heuristic information, their lack of design automation and their unproven ability to reach an optimal solution by linguistic rules alone. Our use of the LMS learning algorithm attempts to solve such problems. The learning algorithm did offer small, incremental improvements, but we believe that the learning algorithm has not yet been explored fully. A better understanding of the learning dynamics would offer more insight into improving the system.

In future works, one may also consider other strategies which use irrelevant questions, (see Fig.7). These questions could be easily exploited for normalizing the data and making it independent of individual charateristics of the tested subjects.

5.2. The polygraph examination:

As expected[61], and eventually proven[62], our clustering system can provide an up to 12% more correct detection rate by using the dependency between the polygraph sessions. Therefore, I recommend recording at least three - ideally five - test sessions with different a order of questions per each examinations. Thus, in cases where some sessions within an examination are clustered incorrectly, the algorithm can easily ignore the minority and find the right cluster according to the correctly clustered majority.

One may also consider other time frames, and emphasize those features which enabled us to cluster the data the best. It may also be helpful to mark the data of female and male subjects, or to consider them differently, since the ranges of the biophysical reactions are not in the same numerical spaces.

Ultimately, an automated polygraph system which uses the aforementioned strategies to distinguish between truth and deception should have a built-in feature extraction tool which can directly feed the needed data to the algorithm.

---

[61]See chapter 3.1.3.4.
[62]See chapter 4.1.2.3.

Feat-ure	Chan-nel	Extraction Method	Combination Method
1	GSR	mean	ave(r) - ave(c)
2	GSR	mean	ave(r) + ave(c)
3	GSR	mean	max(r) - max(c)
4	GSR	mean	min(r) - min(c)
5	GSR	mean	max(r) - min(c)
6	GSR	mean	min(r) - max(c)
7	GSR	curve length	max(r) / max(c)
8	GSR	curve length	ave(r) - ave(c)
9	GSR	curve length	ave(r) + ave(c)
10	GSR	curve length	max(r) - max(c)
11	GSR	curve length	min(r) - min(c)
12	GSR	curve length	max(r) - min(c)
13	GSR	curve length	min(r) - max(c)
14	GSR	area	max(r) / max(c)
15	GSR	area	ave(r) - ave(c)
16	GSR	area	ave(r) + ave(c)
17	GSR	area	max(r) - max(c)
18	GSR	area	min(r) - min(c)
19	GSR	area	max(r) - min(c)
20	GSR	area	min(r) - max(c)
21	GSR	area	max(r) / max(c)
22	GSR	median of the derivative	ave(r) - ave(c)
23	GSR	median of the derivative	ave(r) + ave(c)
24	GSR	median of the derivative	max(r) - max(c)
25	GSR	median of the derivative	min(r) - min(c)
26	GSR	median of the derivative	max(r) - min(c)
27	GSR	median of the derivative	min(r) - max(c)
28	GSR	median of the derivative	max(r) / max(c)
29	GSR	min subtracted from the max	ave(r) - ave(c)
30	GSR	min subtracted from the max	ave(r) + ave(c)
31	GSR	min subtracted from the max	max(r) - max(c)
32	GSR	min subtracted from the max	min(r) - min(c)
33	GSR	min subtracted from the max	max(r) - min(c)
34	GSR	min subtracted from the max	min(r) - max(c)
35	GSR	min subtracted from the max	max(r) / max(c)
36	GSR	maximum of the signal	ave(r) - ave(c)
37	GSR	maximum of the signal	ave(r) + ave(c)
38	GSR	maximum of the signal	max(r) - max(c)
39	GSR	maximum of the signal	min(r) - min(c)
40	GSR	maximum of the signal	max(r) - min(c)
41	GSR	maximum of the signal	min(r) - max(c)
42	GSR	maximum of the signal	max(r) / max(c)
43	GSR	minimum of the signal	ave(r) - ave(c)
44	GSR	minimum of the signal	ave(r) + ave(c)
45	GSR	minimum of the signal	max(r) - max(c)
46	GSR	minimum of the signal	min(r) - min(c)
47	GSR	minimum of the signal	max(r) - min(c)
48	GSR	minimum of the signal	min(r) - max(c)
49	GSR	minimum of the signal	max(r) / max(c)
50	GSR	mean of derivative	ave(r) - ave(c)
51	GSR	mean of derivative	ave(r) + ave(c)
52	GSR	mean of derivative	max(r) - max(c)
53	GSR	mean of derivative	min(r) - min(c)
54	GSR	mean of derivative	max(r) - min(c)
55	GSR	mean of derivative	min(r) - max(c)
56	GSR	mean of derivative	max(r) / max(c)
57	HFEC	mean	ave(r) - ave(c)
58	HFEC	mean	ave(r) + ave(c)
59	HFEC	mean	max(r) - max(c)
60	HFEC	mean	min(r) - min(c)
61	HFEC	mean	max(r) - min(c)
62	HFEC	mean	min(r) - max(c)
63	HFEC	mean	max(r) / max(c)
64	HFEC	curve length	ave(r) - ave(c)
65	HFEC	curve length	ave(r) + ave(c)
66	HFEC	curve length	max(r) - max(c)
67	HFEC	curve length	min(r) - min(c)
68	HFEC	curve length	max(r) - min(c)
69	HFEC	curve length	min(r) - max(c)
70	HFEC	curve length	max(r) / max(c)
71	HFEC	area	ave(r) - ave(c)
72	HFEC	area	ave(r) + ave(c)
73	HFEC	area	max(r) - max(c)
74	HFEC	area	min(r) - min(c)
75	HFEC	area	max(r) - min(c)
76	HFEC	area	min(r) - max(c)
77	HFEC	area	max(r) / max(c)
78	HFEC	amplitude of the peaks	ave(r) - ave(c)
79	HFEC	amplitude of the peaks	ave(r) + ave(c)
80	HFEC	amplitude of the peaks	max(r) - max(c)
81	HFEC	amplitude of the peaks	min(r) - min(c)
82	HFEC	amplitude of the peaks	max(r) - min(c)
83	HFEC	amplitude of the peaks	min(r) - max(c)
84	HFEC	amplitude of the peaks	max(r) / max(c)
85	HFEC	dampcard	ave(r) - ave(c)
86	HFEC	dampcard	ave(r) + ave(c)
87	HFEC	dampcard	max(r) - max(c)
88	HFEC	dampcard	min(r) - min(c)
89	HFEC	dampcard	max(r) - min(c)
90	HFEC	dampcard	min(r) - max(c)
91	HFEC	dampcard	max(r) / max(c)
92	HFEC	number of peaks in cardio	ave(r) - ave(c)
93	HFEC	number of peaks in cardio	ave(r) + ave(c)
94	HFEC	number of peaks in cardio	max(r) - max(c)
95	HFEC	number of peaks in cardio	min(r) - min(c)
96	HFEC	number of peaks in cardio	max(r) - min(c)
97	HFEC	number of peaks in cardio	min(r) - max(c)
98	HFEC	number of peaks in cardio	max(r) / max(c)
99	HFEC	median of the derivative	ave(r) - ave(c)
100	HFEC	median of the derivative	ave(r) + ave(c)
101	HFEC	median of the derivative	max(r) - max(c)
102	HFEC	median of the derivative	min(r) - min(c)
103	HFEC	median of the derivative	max(r) - min(c)
104	HFEC	median of the derivative	min(r) - max(c)
105	HFEC	median of the derivative	max(r) / max(c)
106	HFEC	min subtracted from the max	ave(r) - ave(c)
107	HFEC	min subtracted from the max	ave(r) + ave(c)
108	HFEC	min subtracted from the max	max(r) - max(c)
109	HFEC	min subtracted from the max	min(r) - min(c)
110	HFEC	min subtracted from the max	max(r) - min(c)
111	HFEC	min subtracted from the max	min(r) - max(c)
112	HFEC	min subtracted from the max	max(r) / max(c)
113	HFEC	maximum	ave(r) - ave(c)
114	HFEC	maximum	ave(r) + ave(c)
115	HFEC	maximum	max(r) - max(c)
116	HFEC	maximum	min(r) - min(c)
117	HFEC	maximum	max(r) - min(c)
118	HFEC	maximum	min(r) - max(c)
119	HFEC	maximum	max(r) / max(c)
120	HFEC	minimum	ave(r) - ave(c)
121	HFEC	minimum	ave(r) + ave(c)
122	HFEC	minimum	max(r) - max(c)
123	HFEC	minimum	min(r) - min(c)
124	HFEC	minimum	max(r) - min(c)
125	HFEC	minimum	min(r) - max(c)
126	HFEC	minimum	max(r) / max(c)
127	HFEC	median of the derivative	ave(r) - ave(c)
128	HFEC	median of the derivative	ave(r) + ave(c)
129	HFEC	median of the derivative	max(r) - max(c)
130	HFEC	median of the derivative	min(r) - min(c)
131	HFEC	median of the derivative	max(r) - min(c)
132	HFEC	median of the derivative	min(r) - max(c)
133	HFEC	median of the derivative	max(r) / max(c)
134	HFEC	minampc	ave(r) - ave(c)
135	HFEC	minampc	ave(r) + ave(c)
136	HFEC	minampc	max(r) - max(c)
137	HFEC	minampc	min(r) - min(c)
138	HFEC	minampc	max(r) - min(c)
139	HFEC	minampc	min(r) - max(c)
140	HFEC	minampc	max(r) / max(c)

Fig.41: List of labels of all the features used in this project

141	LC	mean	ave(r) - ave(c)
142	LC	mean	ave(r) + ave(c)
143	LC	mean	max(r) - max(c)
144	LC	mean	min(r) - min(c)
145	LC	mean	max(r) - min(c)
146	LC	mean	min(r) - max(c)
147	LC	mean	max(r) / max(c)
148	LC	curve length	ave(r) - ave(c)
149	LC	curve length	ave(r) + ave(c)
150	LC	curve length	max(r) - max(c)
151	LC	curve length	min(r) - min(c)
152	LC	curve length	max(r) - min(c)
153	LC	curve length	min(r) - max(c)
154	LC	curve length	max(r) / max(c)
155	LC	area	ave(r) - ave(c)
156	LC	area	ave(r) + ave(c)
157	LC	area	max(r) - max(c)
158	LC	area	min(r) - min(c)
159	LC	area	max(r) - min(c)
160	LC	area	min(r) - max(c)
161	LC	area	max(r) / max(c)
162	LC	median of the derivative	ave(r) - ave(c)
163	LC	median of the derivative	ave(r) + ave(c)
164	LC	median of the derivative	max(r) - max(c)
165	LC	median of the derivative	min(r) - min(c)
166	LC	median of the derivative	max(r) - min(c)
167	LC	median of the derivative	min(r) - max(c)
168	LC	median of the derivative	max(r) / max(c)
169	LC	min subtracted from the max	ave(r) - ave(c)
170	LC	min subtracted from the max	ave(r) + ave(c)
171	LC	min subtracted from the max	max(r) - max(c)
172	LC	min subtracted from the max	min(r) - min(c)
173	LC	min subtracted from the max	max(r) - min(c)
174	LC	min subtracted from the max	min(r) - max(c)
175	LC	min subtracted from the max	max(r) / max(c)
176	LC	maximum	ave(r) - ave(c)
177	LC	maximum	ave(r) + ave(c)
178	LC	maximum	max(r) - max(c)
179	LC	maximum	min(r) - min(c)
180	LC	maximum	max(r) - min(c)
181	LC	maximum	min(r) - max(c)
182	LC	maximum	max(r) / max(c)
183	LC	minimum	ave(r) - ave(c)
184	LC	minimum	ave(r) + ave(c)
185	LC	minimum	max(r) - max(c)
186	LC	minimum	min(r) - min(c)
187	LC	minimum	max(r) - min(c)
188	LC	minimum	min(r) - max(c)
189	LC	minimum	max(r) / max(c)
190	LC	median of the derivative	ave(r) - ave(c)
191	LC	median of the derivative	ave(r) + ave(c)
192	LC	median of the derivative	max(r) - max(c)
193	LC	median of the derivative	min(r) - min(c)
194	LC	median of the derivative	max(r) - min(c)
195	LC	median of the derivative	min(r) - max(c)
196	LC	median of the derivative	max(r) / max(c)
197	DLC	mean	ave(r) - ave(c)
198	DLC	mean	ave(r) + ave(c)
199	DLC	mean	max(r) - max(c)
200	DLC	mean	min(r) - min(c)
201	DLC	mean	max(r) - min(c)
202	DLC	mean	min(r) - max(c)
203	DLC	mean	max(r) / max(c)
204	DLC	curve length	ave(r) - ave(c)
205	DLC	curve length	ave(r) + ave(c)
206	DLC	curve length	max(r) - max(c)
207	DLC	curve length	min(r) - min(c)
208	DLC	curve length	max(r) - min(c)
209	DLC	curve length	min(r) - max(c)
210	DLC	curve length	max(r) / max(c)
211	DLC	area	ave(r) - ave(c)
212	DLC	area	ave(r) + ave(c)
213	DLC	area	max(r) - max(c)
214	DLC	area	min(r) - min(c)
215	DLC	area	max(r) - min(c)
216	DLC	area	min(r) - max(c)
217	DLC	area	max(r) / max(c)
218	DLC	median of the derivative	ave(r) - ave(c)
219	DLC	median of the derivative	ave(r) + ave(c)
220	DLC	median of the derivative	max(r) - max(c)
221	DLC	median of the derivative	min(r) - min(c)
222	DLC	median of the derivative	max(r) - min(c)
223	DLC	median of the derivative	min(r) - max(c)
224	DLC	median of the derivative	max(r) / max(c)
225	DLC	min subtracted from the max	ave(r) - ave(c)
226	DLC	min subtracted from the max	ave(r) + ave(c)
227	DLC	min subtracted from the max	max(r) - max(c)
228	DLC	min subtracted from the max	min(r) - min(c)
229	DLC	min subtracted from the max	max(r) - min(c)
230	DLC	min subtracted from the max	min(r) - max(c)
231	DLC	min subtracted from the max	max(r) / max(c)
232	DLC	maximum	ave(r) - ave(c)
233	DLC	maximum	ave(r) + ave(c)
234	DLC	maximum	max(r) - max(c)
235	DLC	maximum	min(r) - min(c)
236	DLC	maximum	max(r) - min(c)
237	DLC	maximum	min(r) - max(c)
238	DLC	maximum	max(r) / max(c)
239	DLC	minimum	ave(r) - ave(c)
240	DLC	minimum	ave(r) + ave(c)
241	DLC	minimum	max(r) - max(c)
242	DLC	minimum	min(r) - min(c)
243	DLC	minimum	max(r) - min(c)
244	DLC	minimum	min(r) - max(c)
245	DLC	minimum	max(r) / max(c)
246	DLC	mean of derivative	ave(r) - ave(c)
247	DLC	mean of derivative	ave(r) + ave(c)
248	DLC	mean of derivative	max(r) - max(c)
249	DLC	mean of derivative	min(r) - min(c)
250	DLC	mean of derivative	max(r) - min(c)
251	DLC	mean of derivative	min(r) - max(c)
252	DLC	mean of derivative	max(r) / max(c)
253	LR	mean	ave(r) - ave(c)
254	LR	mean	ave(r) + ave(c)
255	LR	mean	max(r) - max(c)
256	LR	mean	min(r) - min(c)
257	LR	mean	max(r) - min(c)
258	LR	mean	min(r) - max(c)
259	LR	mean	max(r) / max(c)
260	LR	curve length	ave(r) - ave(c)
261	LR	curve length	ave(r) + ave(c)
262	LR	curve length	max(r) - max(c)
263	LR	curve length	min(r) - min(c)
264	LR	curve length	max(r) - min(c)
265	LR	curve length	min(r) - max(c)
266	LR	curve length	max(r) / max(c)
267	LR	area	ave(r) - ave(c)
268	LR	area	ave(r) + ave(c)
269	LR	area	max(r) - max(c)
270	LR	area	min(r) - min(c)
271	LR	area	max(r) - min(c)
272	LR	area	min(r) - max(c)
273	LR	area	max(r) / max(c)
274	LR	amplitude of the peaks	ave(r) - ave(c)
275	LR	amplitude of the peaks	ave(r) + ave(c)
276	LR	amplitude of the peaks	max(r) - max(c)
277	LR	amplitude of the peaks	min(r) - min(c)
278	LR	amplitude of the peaks	max(r) - min(c)
279	LR	amplitude of the peaks	min(r) - max(c)
280	LR	amplitude of the peaks	max(r) / max(c)

Fig.41: Continued

281	LR	number of the peaks	ave(r) - ave(c)
282	LR	number of the peaks	ave(r) + ave(c)
283	LR	number of the peaks	max(r) - max(c)
284	LR	number of the peaks	min(r) - min(c)
285	LR	number of the peaks	max(r) - min(c)
286	LR	number of the peaks	min(r) - max(c)
287	LR	number of the peaks	max(r) / max(c)
288	LR	inhal_divided_by_exhal	ave(r) - ave(c)
289	LR	inhal_divided_by_exhal	ave(r) + ave(c)
290	LR	inhal_divided_by_exhal	max(r) - max(c)
291	LR	inhal_divided_by_exhal	min(r) - min(c)
292	LR	inhal_divided_by_exhal	max(r) - min(c)
293	LR	inhal_divided_by_exhal	min(r) - max(c)
294	LR	inhal_divided_by_exhal	max(r) / max(c)
295	LR	dampr	ave(r) - ave(c)
296	LR	dampr	ave(r) + ave(c)
297	LR	dampr	max(r) - max(c)
298	LR	dampr	min(r) - min(c)
299	LR	dampr	max(r) - min(c)
300	LR	dampr	min(r) - max(c)
301	LR	dampr	max(r) / max(c)
302	LR	ieie	ave(r) - ave(c)
303	LR	ieie	ave(r) + ave(c)
304	LR	ieie	max(r) - max(c)
305	LR	ieie	min(r) - min(c)
306	LR	ieie	max(r) - min(c)
307	LR	ieie	min(r) - max(c)
308	LR	ieie	max(r) / max(c)
309	LR	median of the derivative	ave(r) - ave(c)
310	LR	median of the derivative	ave(r) + ave(c)
311	LR	median of the derivative	max(r) - max(c)
312	LR	median of the derivative	min(r) - min(c)
313	LR	median of the derivative	max(r) - min(c)
314	LR	median of the derivative	min(r) - max(c)
315	LR	median of the derivative	max(r) / max(c)
316	LR	min subtracted from the max	ave(r) - ave(c)
317	LR	min subtracted from the max	ave(r) + ave(c)
318	LR	min subtracted from the max	max(r) - max(c)
319	LR	min subtracted from the max	min(r) - min(c)
320	LR	min subtracted from the max	max(r) - min(c)
321	LR	min subtracted from the max	min(r) - max(c)
322	LR	min subtracted from the max	max(r) / max(c)
323	LR	maximum	ave(r) - ave(c)
324	LR	maximum	ave(r) + ave(c)
325	LR	maximum	max(r) - max(c)
326	LR	maximum	min(r) - min(c)
327	LR	maximum	max(r) - min(c)
328	LR	maximum	min(r) - max(c)
329	LR	maximum	max(r) / max(c)
330	LR	minimum	ave(r) - ave(c)
331	LR	minimum	ave(r) + ave(c)
332	LR	minimum	max(r) - max(c)
333	LR	minimum	min(r) - min(c)
334	LR	minimum	max(r) - min(c)
335	LR	minimum	min(r) - max(c)
336	LR	minimum	max(r) / max(c)
337	LR	mean of derivative	ave(r) - ave(c)
338	LR	mean of derivative	ave(r) + ave(c)
339	LR	mean of derivative	max(r) - max(c)
340	LR	mean of derivative	min(r) - min(c)
341	LR	mean of derivative	max(r) - min(c)
342	LR	mean of derivative	min(r) - max(c)
343	LR	mean of derivative	max(r) / max(c)
344	LR	minampr	ave(r) - ave(c)
345	LR	minampr	ave(r) + ave(c)
346	LR	minampr	max(r) - max(c)
347	LR	minampr	min(r) - min(c)
348	LR	minampr	max(r) - min(c)
349	LR	minampr	min(r) - max(c)
350	LR	minampr	max(r) / max(c)

351	UR	mean	ave(r) - ave(c)
352	UR	mean	ave(r) + ave(c)
353	UR	mean	max(r) - max(c)
354	UR	mean	min(r) - min(c)
355	UR	mean	max(r) - min(c)
356	UR	mean	min(r) - max(c)
357	UR	mean	max(r) / max(c)
358	UR	curve length	ave(r) - ave(c)
359	UR	curve length	ave(r) + ave(c)
360	UR	curve length	max(r) - max(c)
361	UR	curve length	min(r) - min(c)
362	UR	curve length	max(r) - min(c)
363	UR	curve length	min(r) - max(c)
364	UR	curve length	max(r) / max(c)
365	UR	area	ave(r) - ave(c)
366	UR	area	ave(r) + ave(c)
367	UR	area	max(r) - max(c)
368	UR	area	min(r) - min(c)
369	UR	area	max(r) - min(c)
370	UR	area	min(r) - max(c)
371	UR	area	max(r) / max(c)
372	UR	amplitude of the peaks	ave(r) - ave(c)
373	UR	amplitude of the peaks	ave(r) + ave(c)
374	UR	amplitude of the peaks	max(r) - max(c)
375	UR	amplitude of the peaks	min(r) - min(c)
376	UR	amplitude of the peaks	max(r) - min(c)
377	UR	amplitude of the peaks	min(r) - max(c)
378	UR	amplitude of the peaks	max(r) / max(c)
379	UR	dampr	ave(r) - ave(c)
380	UR	dampr	ave(r) + ave(c)
381	UR	dampr	max(r) - max(c)
382	UR	dampr	min(r) - min(c)
383	UR	dampr	max(r) - min(c)
384	UR	dampr	min(r) - max(c)
385	UR	dampr	max(r) / max(c)
386	UR	number of the peaks	ave(r) - ave(c)
387	UR	number of the peaks	ave(r) + ave(c)
388	UR	number of the peaks	max(r) - max(c)
389	UR	number of the peaks	min(r) - min(c)
390	UR	number of the peaks	max(r) - min(c)
391	UR	number of the peaks	min(r) - max(c)
392	UR	number of the peaks	max(r) / max(c)
393	UR	inhal_divided_by_exhal	ave(r) - ave(c)
394	UR	inhal_divided_by_exhal	ave(r) + ave(c)
395	UR	inhal_divided_by_exhal	max(r) - max(c)
396	UR	inhal_divided_by_exhal	min(r) - min(c)
397	UR	inhal_divided_by_exhal	max(r) - min(c)
398	UR	inhal_divided_by_exhal	min(r) - max(c)
399	UR	inhal_divided_by_exhal	max(r) / max(c)
400	UR	ieie	ave(r) - ave(c)
401	UR	ieie	ave(r) + ave(c)
402	UR	ieie	max(r) - max(c)
403	UR	ieie	min(r) - min(c)
404	UR	ieie	max(r) - min(c)
405	UR	ieie	min(r) - max(c)
406	UR	ieie	max(r) / max(c)
407	UR	median of the derivative	ave(r) - ave(c)
408	UR	median of the derivative	ave(r) + ave(c)
409	UR	median of the derivative	max(r) - max(c)
410	UR	median of the derivative	min(r) - min(c)
411	UR	median of the derivative	max(r) - min(c)
412	UR	median of the derivative	min(r) - max(c)
413	UR	median of the derivative	max(r) / max(c)
414	UR	min subtracted from the max	ave(r) - ave(c)
415	UR	min subtracted from the max	ave(r) + ave(c)
416	UR	min subtracted from the max	max(r) - max(c)
417	UR	min subtracted from the max	min(r) - min(c)
418	UR	min subtracted from the max	max(r) - min(c)
419	UR	min subtracted from the max	min(r) - max(c)
420	UR	min subtracted from the max	max(r) / max(c)

Fig.41: Continued

421	UR	maximum	ave(r) - ave(c)
422	UR	maximum	ave(r) + ave(c)
423	UR	maximum	max(r) - max(c)
424	UR	maximum	min(r) - min(c)
425	UR	maximum	max(r) - min(c)
426	UR	maximum	min(r) - max(c)
427	UR	maximum	max(r) / max(c)
428	UR	minimum	ave(r) - ave(c)
429	UR	minimum	ave(r) + ave(c)
430	UR	minimum	max(r) - max(c)
431	UR	minimum	min(r) - min(c)
432	UR	minimum	max(r) - min(c)
433	UR	minimum	min(r) - max(c)
434	UR	minimum	max(r) / max(c)
435	UR	mean of derivative	ave(r) - ave(c)
436	UR	mean of derivative	ave(r) + ave(c)
437	UR	mean of derivative	max(r) - max(c)
438	UR	mean of derivative	min(r) - min(c)
439	UR	mean of derivative	max(r) - min(c)
440	UR	mean of derivative	min(r) - max(c)
441	UR	mean of derivative	max(r) / max(c)
442	UR	minampr	ave(r) - ave(c)
443	UR	minampr	ave(r) + ave(c)
444	UR	minampr	max(r) - max(c)
445	UR	minampr	min(r) - min(c)
446	UR	minampr	max(r) - min(c)
447	UR	minampr	min(r) - max(c)
448	UR	minampr	max(r) / max(c)
449	GSR	standard deviation	ave(r) - ave(c)
450	GSR	standard deviation	ave(r) + ave(c)
451	GSR	standard deviation	max(r) - max(c)
452	GSR	standard deviation	min(r) - min(c)
453	GSR	standard deviation	max(r) - min(c)
454	GSR	standard deviation	min(r) - max(c)
455	GSR	standard deviation	max(r) / max(c)
456	HFEC	standard deviation	ave(r) - ave(c)
457	HFEC	standard deviation	ave(r) + ave(c)
458	HFEC	standard deviation	max(r) - max(c)
459	HFEC	standard deviation	min(r) - min(c)
460	HFEC	standard deviation	max(r) - min(c)
461	HFEC	standard deviation	min(r) - max(c)
462	HFEC	standard deviation	max(r) / max(c)
463	LC	standard deviation	ave(r) - ave(c)
464	LC	standard deviation	ave(r) + ave(c)
465	LC	standard deviation	max(r) - max(c)
466	LC	standard deviation	min(r) - min(c)
467	LC	standard deviation	max(r) - min(c)
468	LC	standard deviation	min(r) - max(c)
469	LC	standard deviation	max(r) / max(c)
470	DLC	standard deviation	ave(r) - ave(c)
471	DLC	standard deviation	ave(r) + ave(c)
472	DLC	standard deviation	max(r) - max(c)
473	DLC	standard deviation	min(r) - min(c)
474	DLC	standard deviation	max(r) - min(c)
475	DLC	standard deviation	min(r) - max(c)
476	DLC	standard deviation	max(r) / max(c)
477	LR	standard deviation	ave(r) - ave(c)
478	LR	standard deviation	ave(r) + ave(c)
479	LR	standard deviation	max(r) - max(c)
480	LR	standard deviation	min(r) - min(c)
481	LR	standard deviation	max(r) - min(c)
482	LR	standard deviation	min(r) - max(c)
483	LR	standard deviation	max(r) / max(c)
484	UR	standard deviation	ave(r) - ave(c)
485	UR	standard deviation	ave(r) + ave(c)
486	UR	standard deviation	max(r) - max(c)
487	UR	standard deviation	min(r) - min(c)
488	UR	standard deviation	max(r) - min(c)
489	UR	standard deviation	min(r) - max(c)
490	UR	standard deviation	max(r) / max(c)

491	HFEC	coeff_of_ARmod	ave(r) - ave(c)
492	HFEC	coeff_of_ARmod	ave(r) + ave(c)
493	HFEC	coeff_of_ARmod	max(r) - max(c)
494	HFEC	coeff_of_ARmod	min(r) - min(c)
495	HFEC	coeff_of_ARmod	max(r) - min(c)
496	HFEC	coeff_of_ARmod	min(r) - max(c)
497	HFEC	coeff_of_ARmod	max(r) / max(c)
498	HFEC	coeff_of_ARmod	ave(r) - ave(c)
499	HFEC	coeff_of_ARmod	ave(r) + ave(c)
500	HFEC	coeff_of_ARmod	max(r) - max(c)
501	HFEC	coeff_of_ARmod	min(r) - min(c)
502	HFEC	coeff_of_ARmod	max(r) - min(c)
503	HFEC	coeff_of_ARmod	min(r) - max(c)
504	HFEC	coeff_of_ARmod	max(r) / max(c)
505	HFEC	coeff_of_ARmod	ave(r) - ave(c)
506	HFEC	coeff_of_ARmod	ave(r) + ave(c)
507	HFEC	coeff_of_ARmod	max(r) - max(c)
508	HFEC	coeff_of_ARmod	min(r) - min(c)
509	HFEC	coeff_of_ARmod	max(r) - min(c)
510	HFEC	coeff_of_ARmod	min(r) - max(c)
511	HFEC	coeff_of_ARmod	max(r) / max(c)
512	HFEC	coeff_of_ARmod	ave(r) - ave(c)
513	HFEC	coeff_of_ARmod	ave(r) + ave(c)
514	HFEC	coeff_of_ARmod	max(r) - max(c)
515	HFEC	coeff_of_ARmod	min(r) - min(c)
516	HFEC	coeff_of_ARmod	max(r) - min(c)
517	HFEC	coeff_of_ARmod	min(r) - max(c)
518	HFEC	coeff_of_ARmod	max(r) / max(c)
519	HFEC	coeff_of_ARmod	ave(r) - ave(c)
520	HFEC	coeff_of_ARmod	ave(r) + ave(c)
521	HFEC	coeff_of_ARmod	max(r) - max(c)
522	HFEC	coeff_of_ARmod	min(r) - min(c)
523	HFEC	coeff_of_ARmod	max(r) - min(c)
524	HFEC	coeff_of_ARmod	min(r) - max(c)
525	HFEC	coeff_of_ARmod	max(r) / max(c)
526	HFEC	coeff_of_ARmod	ave(r) - ave(c)
527	HFEC	coeff_of_ARmod	ave(r) + ave(c)
528	HFEC	coeff_of_ARmod	max(r) - max(c)
529	HFEC	coeff_of_ARmod	min(r) - min(c)
530	HFEC	coeff_of_ARmod	max(r) - min(c)
531	HFEC	coeff_of_ARmod	min(r) - max(c)
532	HFEC	coeff_of_ARmod	max(r) / max(c)
533	HFEC	coeff_of_ARmod	ave(r) - ave(c)
534	HFEC	coeff_of_ARmod	ave(r) + ave(c)
535	HFEC	coeff_of_ARmod	max(r) - max(c)
536	HFEC	coeff_of_ARmod	min(r) - min(c)
537	HFEC	coeff_of_ARmod	max(r) - min(c)
538	HFEC	coeff_of_ARmod	min(r) - max(c)
539	HFEC	coeff_of_ARmod	max(r) / max(c)
540	HFEC	coeff_of_ARmod	ave(r) - ave(c)
541	HFEC	coeff_of_ARmod	ave(r) + ave(c)
542	HFEC	coeff_of_ARmod	max(r) - max(c)
543	HFEC	coeff_of_ARmod	min(r) - min(c)
544	HFEC	coeff_of_ARmod	max(r) - min(c)
545	HFEC	coeff_of_ARmod	min(r) - max(c)
546	HFEC	coeff_of_ARmod	max(r) / max(c)
547	HFEC	coeff_of_ARmod	ave(r) - ave(c)
548	HFEC	coeff_of_ARmod	ave(r) + ave(c)
549	HFEC	coeff_of_ARmod	max(r) - max(c)
550	HFEC	coeff_of_ARmod	min(r) - min(c)
551	HFEC	coeff_of_ARmod	max(r) - min(c)
552	HFEC	coeff_of_ARmod	min(r) - max(c)
553	HFEC	coeff_of_ARmod	max(r) / max(c)
554	HFEC	coeff_of_ARmod	ave(r) - ave(c)
555	HFEC	coeff_of_ARmod	ave(r) + ave(c)
556	HFEC	coeff_of_ARmod	max(r) - max(c)
557	HFEC	coeff_of_ARmod	min(r) - min(c)
558	HFEC	coeff_of_ARmod	max(r) - min(c)
559	HFEC	coeff_of_ARmod	min(r) - max(c)
560	HFEC	coeff_of_ARmod	max(r) / max(c)

Fig.41: Continued

561	HFEC	fund_fmax_cross_corr	ave(r) - ave(c)
562	HFEC	fund_fmax_cross_corr	ave(r) + ave(c)
563	HFEC	fund_fmax_cross_corr	max(r) - max(c)
564	HFEC	fund_fmax_cross_corr	min(r) - min(c)
565	HFEC	fund_fmax_cross_corr	max(r) - min(c)
567	HFEC	fund_fmax_cross_corr	min(r) - max(c)
568	LR	fund_fmax_cross_corr	max(r) / max(c)
569	LR	fund_fmax_cross_corr	ave(r) - ave(c)
570	LR	fund_fmax_cross_corr	ave(r) + ave(c)
571	LR	fund_fmax_cross_corr	max(r) - max(c)
572	LR	fund_fmax_cross_corr	min(r) - min(c)
573	LR	fund_fmax_cross_corr	max(r) - min(c)
574	LR	fund_fmax_cross_corr	min(r) - max(c)
575	HFUR	max_cross_correlation	max(r) / max(c)
576	HFUR	max_cross_correlation	ave(r) - ave(c)
577	HFUR	max_cross_correlation	ave(r) + ave(c)
578	HFUR	max_cross_correlation	max(r) - max(c)
579	HFUR	max_cross_correlation	min(r) - min(c)
580	HFUR	max_cross_correlation	max(r) - min(c)
581	HFUR	max_cross_correlation	min(r) - max(c)
582	HFUR	lag_max_cross_correlation	max(r) / max(c)
583	HFUR	lag_max_cross_correlation	ave(r) - ave(c)
584	HFUR	lag_max_cross_correlation	ave(r) + ave(c)
585	HFUR	lag_max_cross_correlation	max(r) - max(c)
586	HFUR	lag_max_cross_correlation	min(r) - min(c)
587	HFUR	lag_max_cross_correlation	max(r) - min(c)
588	HFUR	lag_max_cross_correlation	min(r) - max(c)
589	HFUR	min_cross_correlation	max(r) / max(c)
590	HFUR	min_cross_correlation	ave(r) - ave(c)
591	HFUR	min_cross_correlation	ave(r) + ave(c)
592	HFUR	min_cross_correlation	max(r) - max(c)
593	HFUR	min_cross_correlation	min(r) - min(c)
594	HFUR	min_cross_correlation	max(r) - min(c)
595	HFUR	min_cross_correlation	min(r) - max(c)
596	HFUR	lag_min_cross_correlation	max(r) / max(c)
597	HFUR	lag_min_cross_correlation	ave(r) - ave(c)
598	HFUR	lag_min_cross_correlation	ave(r) + ave(c)
599	HFUR	lag_min_cross_correlation	max(r) - max(c)
600	HFUR	lag_min_cross_correlation	min(r) - min(c)
601	HFUR	lag_min_cross_correlation	max(r) - min(c)
602	HFUR	lag_min_cross_correlation	min(r) - max(c)
603	HFEC	spec_HFEC_fund_freq	max(r) / max(c)
604	HFEC	spec_HFEC_fund_freq	ave(r) - ave(c)
605	HFEC	spec_HFEC_fund_freq	ave(r) + ave(c)
606	HFEC	spec_HFEC_fund_freq	max(r) - max(c)
607	HFEC	spec_HFEC_fund_freq	min(r) - min(c)
608	HFEC	spec_HFEC_fund_freq	max(r) - min(c)
609	HFEC	spec_HFEC_fund_freq	min(r) - max(c)
610	HFEC	spec_HFEC_2nd_harmonic	max(r) / max(c)
611	HFEC	spec_HFEC_2nd_harmonic	ave(r) - ave(c)
612	HFEC	spec_HFEC_2nd_harmonic	ave(r) + ave(c)
613	HFEC	spec_HFEC_2nd_harmonic	max(r) - max(c)
614	HFEC	spec_HFEC_2nd_harmonic	min(r) - min(c)
615	HFEC	spec_HFEC_2nd_harmonic	max(r) - min(c)
616	HFEC	spec_HFEC_2nd_harmonic	min(r) - max(c)
617	UR	spec_UR_fund_frequency	max(r) / max(c)
618	UR	spec_UR_fund_frequency	ave(r) - ave(c)
619	UR	spec_UR_fund_frequency	ave(r) + ave(c)
620	UR	spec_UR_fund_frequency	max(r) - max(c)
621	UR	spec_UR_fund_frequency	min(r) - min(c)
622	UR	spec_UR_fund_frequency	max(r) - min(c)
623	UR	spec_UR_fund_frequency	min(r) - max(c)
624	UR	spec_UR_2nd_harmonic	max(r) / max(c)
625	UR	spec_UR_2nd_harmonic	ave(r) - ave(c)
626	UR	spec_UR_2nd_harmonic	ave(r) + ave(c)
627	UR	spec_UR_2nd_harmonic	max(r) - max(c)
628	UR	spec_UR_2nd_harmonic6	min(r) - min(c)
629	UR	spec_UR_2nd_harmonic	max(r) - min(c)
630	UR	spec_UR_2nd_harmonic	min(r) - max(c)

631	HFUR	max_cross_spec_density	max(r) / max(c)
632	HFUR	max_cross_spec_density	ave(r) - ave(c)
633	HFUR	max_cross_spec_density	ave(r) + ave(c)
634	HFUR	max_cross_spec_density	max(r) - max(c)
635	HFUR	max_cross_spec_density	min(r) - min(c)
636	HFUR	max_cross_spec_density	max(r) - min(c)
637	HFUR	max_cross_spec_density	min(r) - max(c)
638	HFEC	coherency_HFEC & UR ff	max(r) / max(c)
639	HFEC	coherency_HFEC & UR ff	ave(r) - ave(c)
640	HFEC	coherency_HFEC & UR ff	ave(r) + ave(c)
641	HFEC	coherency_HFEC & UR ff	max(r) - max(c)
642	HFEC	coherency_HFEC & UR ff	min(r) - min(c)
643	HFEC	coherency_HFEC & UR ff	max(r) - min(c)
644	HFEC	coherency_HFEC & UR ff	min(r) - max(c)
645	HFEC	coherency_HFEC & UR sh	max(r) / max(c)
646	HFEC	coherency_HFEC & UR sh	ave(r) - ave(c)
647	HFEC	coherency_HFEC & UR sh	ave(r) + ave(c)
648	HFEC	coherency_HFEC & UR sh	max(r) - max(c)
649	HFEC	coherency_HFEC & UR sh	min(r) - min(c)
650	HFEC	coherency_HFEC & UR sh	max(r) - min(c)
651	HFEC	coherency_HFEC & UR sh	min(r) - max(c)
652	GSR	max_min_ISD_cont & relv	mean(r & c)
653	GSR	max_min_ISD_cont & relv	max(r & c)
654	GSR	max_min_ISD_cont & relv	min(r & c)
655	GSR	freq_max_ISD	mean(r & c)
656	GSR	freq_max_ISD	max(r & c)
657	GSR	freq_max_ISD	min(r & c)
658	GSR	area_under_ISD	mean(r & c)
659	GSR	area_under_ISD	max(r & c)
660	GSR	area_under_ISD	min(r & c)
661	HFEC	max_min_ISD	mean(r & c)
662	HFEC	max_min_ISD	max(r & c)
663	HFEC	max_min_ISD	min(r & c)
664	HFEC	freq_max_ISD	mean(r & c)
665	HFEC	freq_max_ISD	max(r & c)
666	HFEC	freq_max_ISD	min(r & c)
667	HFEC	area_under_ISD	mean(r & c)
668	HFEC	area_under_ISD	max(r & c)
669	HFEC	area_under_ISD	min(r & c)

Non-deceptive	Deceptive 1	Deceptive 2	Deceptive 3
QQ8R9OIO.011	QQ4Q1O83.011	QQ7LX5Q0.021	QQ8RAJ0C.011
QQ8R9OIO.021	QQ4Q1O83.021	QQ7LX5Q0.031	QQ8RAJ0C.021
QQ8R9OIO.031	QQ4Q1O83.031	QQ7MN2Y0.011	QQ8RAJ0C.031
QQ95LU1T.011	QQ4Q3MDC.011	QQ7MN2Y0.021	QQ9EUKVT.011
QQ95LU1T.021	QQ4Q3MDC.021	QQ7MN2Y0.031	QQ9EUKVT.021
QQ95LU1T.031	QQ4Q3MDC.031	QQ7TC5UF.011	QQ9EUKVT.031
QQAURNUS.021	QQ51DE36.011	QQ7TC5UF.021	QQ9IOOXO.021
QQAURNUS.031	QQ51DE36.021	QQ7TC5UF.031	QQ9IOOXO.041
QQAV53P6.011	QQ51DE36.041	QQ7TQVER.011	QQ9SOW8L.011
QQAV53P6.021	QQ6RQGH6.011	QQ7TQVER.021	QQ9SOW8L.021
QQAV53P6.031	QQ6RQGH6.021	QQ7TQVER.031	QQ9SOW8L.031
QQBQ4SHI.011	QQ6RQGH6.031	QQ7TVADC.011	QQ9SQIK9.011
QQBQ4SHI.021	QQ6RQGH6.041	QQ7TVADC.021	QQ9SQIK9.021
QQBQ4SHI.031	QQ6T711O.011	QQ7TVADC.031	QQ9SQIK9.031
QQBSS7WT.011	QQ6T711O.021	QQ7U2T4R.011	QQ9W0B9F.011
QQBSS7WT.021	QQ6T711O.031	QQ7U2T4R.021	QQ9W0B9F.031
QQBSS7WT.031	QQ6Z59IG.011	QQ7U2T4R.031	QQ9W0B9F.041
QQ7OXM60.021	QQ6Z59IG.021	QQ7YP7QU.011	QQ9U4FMU.011
QQ7RH0RO.011	QQ6Z59IG.031	QQ7YP7QU.021	QQ9U4FMU.021
QQ7RH0RO.021	QQ7PP9B9.011	QQ7YP7QU.031	QQ9U4FMU.031
QQ7RH0RO.031	QQ7PP9B9.021	QQ7YZOJ3.011	QQ9Y_SVF.011
QQ7R51P9.011	QQ7PP9B9.031	QQ7YZOJ3.021	QQ9Y_SVF.021
QQ7R51P9.021	QQ7PDU1X.011	QQ7YZOJ3.031	QQ9Y_SVF.031
QQ7R51P9.031	QQ7PDU1X.021	QQ8_0DPT.011	QQ9YH3QF.011
QQ9TDSP3.011	QQ7PDU1X.031	QQ8_0DPT.021	QQ9YH3QF.021
QQ9TDSP3.021	QQ7_PIPF.011	QQ8_0DPT.031	QQ9YH3QF.031
QQ9TDSP3.031	QQ7_PIPF.021	QQ8_0DPT.041	QQA2TT4C.011
QQA8OWOI.011	QQ7_PIPF.031	QQ8_2UQ9.011	QQA2TT4C.021
QQA8OWOI.021	QQ7_JT70.011	QQ8_2UQ9.021	QQA2TT4C.031
QQA8OWOI.031	QQ7_JT70.021	QQ8_2UQ9.031	QQA3HIRX.011
QQBT22O6.011	QQ7_JT70.031	QQ800IG6.011	QQA3HIRX.021
QQBT22O6.021	QQ738DYX.011	QQ800IG6.021	QQA3HIRX.031
QQBT22O6.031	QQ738DYX.021	QQ800IG6.031	QQA32UTF.011
QQBO9O_9.011	QQ738DYX.031	QQ82OIU9.011	QQA32UTF.021
QQBO9O_9.021	QQ75ULP9.011	QQ82OIU9.021	QQA32UTF.031
QQBO9O_9.031	QQ75ULP9.021	QQ82OIU9.031	QQA6U_IF.011
QQBC7PP6.011	QQ75ULP9.031	QQ82SUTX.011	QQA6U_IF.031
QQBC7PP6.021	QQ79_EYF.011	QQ82SUTX.021	QQA6U_IF.041
QQBC7PP6.031	QQ79_EYF.021	QQ82SUTX.031	QQAM4E3L.011
QQCHCK_O.011	QQ79_EYF.031	QQ860ZNU.011	QQAM4E3L.021
QQCHCK_O.021	QQ7BGDML.011	QQ860ZNU.021	QQAM4E3L.031
QQCHCK_O.031	QQ7BGDML.021	QQ860ZNU.031	QQARF2_X.011
QQCDTKP0.011	QQ7BGDML.031	QQ89U_ZR.011	QQARF2_X.021
QQCDTKP0.031	QQ7ETC8I.011	QQ89U_ZR.021	QQARF2_X.031
QQCDTKP0.041	QQ7ETC8I.021	QQ89U_ZR.031	QQAWA38X.011
QQCM5Y56.011	QQ7ETC8I.031	QQ8ATU26.011	QQAWA38X.021
QQCQQT8Y.011	QQ7JAQCS.011	QQ8ATU26.021	QQAWA38X.031
QQCQQT8Y.021	QQ7JAQCS.021	QQ8ATU26.031	QQAYXZGU.011
QQCQQT8Y.031	QQ7JAQCS.031	QQ8FGMVI.011	QQAYXZGU.021
QQCQQT8Y.041	QQ7LX5Q0.011	QQ8FGMVI.021	QQAYXZGU.031

Fig.42: List of polygraph files used in this experiment

6.3. USER INTERFACE

For an automated polygraph system as a real product, the existence of an user-friendly interface is unavoidable. MATLAB software environment provide an easy-to-use toolbox for creating various kinds of interactive interface classes. The following figure shows an interface used in one of my representations. This was made for a technically oriented user who is familiar with the algorithm. A simpler black-box version of a polygraph system, appropriate to the user's requests, can likewise be programmed.

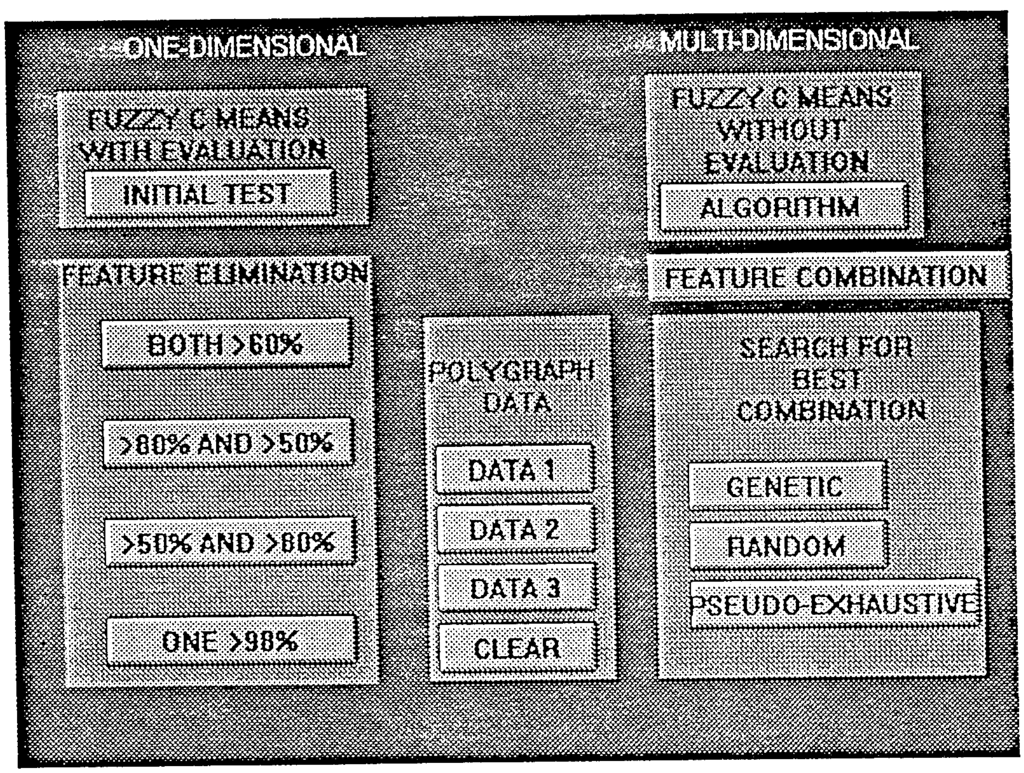


Fig.43: An example for a *technical user* interface

6.4. PROGRAM LISTINGS
(Implementation in MATLAB)

```
% THIS PROGRAM CALCULATESTHE CLUSTER CENTERS FOR
% A MULTIDIMENSIONAL FCM - C=2, CONST.

function V = c_center(X, U, m)

[colE, rowE] = size(X);
k=1:rowE;

%for the 1th class:

V1_numerator = U(1,k).^m * X(:,k)';
% (.*)==>(*): because the "numerator sum" is automatically
% included within the matrix multiplication.

V(1,:) = V1_numerator / sum( U(1,k).^m );
% V(1,:) [and V1_numerator] is a n-dimensional row-vector;
% n represents the number of the clustering features(n=30).

%for the 2nd class:

V2_numerator = U(2,k).^m * X(:,k)';
% (.*)==>(*): ...see above.

V(2,:) = V2_numerator / sum( U(2,k).^m );
% This is a n-dimensional row-vector and the cluster-center
% of the 2nd class.

V=V';                    % [nxc] matrix
return;

**********************************************************************
% FUZZY C-MEANS ALGORITHM FOR MULTI-DIMENSIONAL FCM.

%function best_Uik = fc_means(m, epsilon,X)
function [best_Uik, z] = fc_means(m, epsilon,X)
%function best_Uik = fc_means(m, epsilon)
%function [best_Uik, V, X] = fc_means(m, epsilon)
% think about the X

load init_u;             % start with the initialization of the memb_fct
                         % (Uik ==> Vi)

% load init_v;           % or with the cluster centers
                         % (Vi ==> Uik)

%load set31;             % including the data X respect. X1, X2, ...
%X=featmat;
%load set3me;X=Xselect;

%format long;            % avoid errors by visual comparing the numbers
J_m = 100000000;         % to make sure the start is o.k.
z=0;
while J_m > epsilon

        V = c_center(X, U, m);

        U = memb_fct(X, V, m);

        Jtemp = J_m;
        J_m = j_mdim(X, V, U, m);

 if epsilon <= 0.000005
        if (abs(J_m - Jtemp) <= .00000000001),
        %if J_m == Jtemp,          % to terminate the loop by reaching
                                   % the minimum of J_m.
         break,
        end
 else
        if (abs(J_m - Jtemp) <= .0001),%---o.k.
        %if J_m == Jtemp,          % to terminate the loop by reaching
                                   % the minimum of J_m.
         break,
        end

 end

 %        t = abs(U - temp);       % tolerance value for the iteration

 z=z+1;
  if rem(z,10) ==0
   fprintf('.\n');
  else
   fprintf('.');
  end
 end

  fprintf('\n');fprintf('_________________\n');

 best_Uik = U;
```

```
%Vnew = V;

% recall the extrem values: J_m =7.2308e+003

return;
*********************************************************************
% THIS PROGRAM CALCULATES THE OBJECTIVE FUNCTION
% FOR THE MULTIDIMENSIONAL FCM.

function J_m = j_mdim(X, V, U, m)

[colE,rowE] = size(X);
k = 1:rowE;

%for the 1th class:
V1asMatrix = V(:,1)*ones(1,rowE);                   % to avoid time-crunching for-loops

temp1 = ( X(:,k) - V1asMatrix )' * ( X(:,k) - V1asMatrix );   % trick: matrix-operation is faster,the sought norm is
                                                     % automatically the diagonal of temp1;

temp11 = (  (U(1,:).^m) .* (diag(temp1)')  );

J_out1 = sum(temp11);


%for the 2nd class:
V2asMatrix = V(:,2)*ones(1,rowE);                   % to avoid time-crunching for-loops

temp2 = ( X(:,k) - V2asMatrix )' * ( X(:,k) - V2asMatrix );   % see above

temp22 = (  (U(2,:).^m) .* (diag(temp2)')  );

J_out2 = sum(temp22);

J_m = J_out1 + J_out2;
return;
*********************************************************************
% THIS PROGRAM CALCULATES THE MEMBERSHIP VALUES FOR
% THE MULTIDIMENSIONAL FCM.

function U = memb_fct(X, V, m)

[colE,rowE] = size(X);
k = 1:rowE;

%for the 1th class:

V1asMatrix = V(:,1)*ones(1,rowE);
             % to avoid time-crunching for-loops

temp1 = ( X(:,k) - V1asMatrix )' * ( X(:,k) - V1asMatrix );
             % trick: matrix-operation is faster,the sought norm is
             % automatically the diagonal of temp1;

U_num(1,k) = ( diag(temp1)' ) .^ ( -1/(m-1) );

%for the 2nd class:

V2asMatrix = V(:,2)*ones(1,rowE);
             % to avoid time-crunching for-loops

temp2 = ( X(:,k) - V2asMatrix )' * ( X(:,k) - V2asMatrix );
             % see above

U_num(2,k) = ( diag(temp2)' ) .^ ( -1/(m-1) );


U(1,:) = U_num(1,k) ./ (  U_num(1,k) + U_num(2,k)  );
U(2,:) = U_num(2,k) ./ (  U_num(1,k) + U_num(2,k)  );

% If there is a third class, " U_num(3,k) ... "
% must be also considered.

return;
*********************************************************************
% FAST MULTIDIMENSIONAL EVALUATING PROGRAM
clear best_Uik;

%------without plots

best_Uik = fc_means(5, 0.0000005, Xselect);


figure(1);clg;hold on;
ss=1:100;
plot(ss,best_Uik(1,:),'+');plot(ss,best_Uik(2,:),'*b');
%plot(ss,best_Uik(3,:),'*b')
pause;
```

```
wrong_dcps = 0;
wrong_nons = 0;
figure(2);clg;hold on;
for s=1:100
        if best_Uik(2,s)>=.5
         plot(s,best_Uik(2,s),'*b');
                if s>50
                wrong_dcps=wrong_dcps+1;
                end
        else
         plot(s,best_Uik(2,s),'+');
                if s<=50
                wrong_nons=wrong_nons+1;
                end
        end
end

wpercent = wrong_dcps/50*100;
%fprintf('wrong_dcps, percent')
%[wrong_dcps, wpercent]
npercent = wrong_nons/50*100;
%fprintf('wrong_nons, npercent')
%[wrong_nons, npercent]

nn=(100-npercent);
ww=(100-wpercent);

fprintf('\n');fprintf('RIGHT DETECTIONS:');
fprintf('\n');fprintf('\n');fprintf('nD-clust D_clust');

[nn ww],
**********************************************************************

% USER INTERFACE
% Program B1. This program creates the start button.

figure(1);clg;
set(gcf,'color',[1 0 1])

button1 = uicontrol(gcf,...
'style','push',...
'position',[195 150 75 75],...
'string','START,...
'callback','bt_choic');
**********************************************************************
% USER INTERFACE
%Program B2. This program displays choices to run the various programs.


clf reset
set(gcf,'color',[0 0 1])

title('ONE-DIMENSIONAL                    MULTI-DIMENSIONAL')

axis off

frm2 = uicontrol(gcf,...
'style','text',...
'position',[25 40 155 200]);

tt2 = uicontrol(gcf,...
'style','text',...
'string','FEATURE ELIMINATION',...
'position',[25 215 155 40]);

frm4 = uicontrol(gcf,...
'style','frame',...
'position',[25 270 155 70]);

tt4 = uicontrol(gcf,...
'style','text',...
'string','FUZZY C MEANS WITH EVALUATION',...
'position',[35 288 125 45]);


button3 = uicontrol(gcf,...
'style','push',...
'position',[38 275 125 25],...
'string','INITIAL TEST,...
'callback','mega_tst');

frm = uicontrol(gcf,...
'style','frame',...
'position',[205 40 95 185]);

tt = uicontrol(gcf,...
'style','text',...
```

```
'string','POLYGRAPH DATA',...
'position',[207 165 85 40]);

button13 = uicontrol(gcf,...
'style','push',...
'position',[210 75 80 25],...
'string','DATA 3',...
'callback','load ftx3');

button14 = uicontrol(gcf,...
'style','push',...
'position',[210 105 80 25],...
'string','DATA 2',...
'callback','load ftx2');

button15 = uicontrol(gcf,...
'style','push',...
'position',[210 135 80 25],...
'string','DATA 1',...
'callback','load ftx1');

button16 = uicontrol(gcf,...
'style','push',...
'position',[210 45 80 25],...
'string','CLEAR',...
'callback','clear');

button17 = uicontrol(gcf,...
'style','push',...
'position',[45 200 125 25],...
'string','BOTH >60%',...
'callback','mega_i');

button18 = uicontrol(gcf,...
'style','push',...
'position',[45 150 125 25],...
'string','>80% AND >50%',...
'callback','mega_ii');

button19 = uicontrol(gcf,...
'style','push',...
'position',[45 100 125 25],...
'string','>50% AND >80%',...
'callback','mega_iii');

button20 = uicontrol(gcf,...
'style','push',...
'position',[45 50 125 25],...
'string','ONE >98%',...
'callback','mega_iv');

frm3 = uicontrol(gcf,...
'style','frame',...
'position',[320 40 165 185]);

tt3 = uicontrol(gcf,...
'style','text',...
'string','SEARCH FOR BEST COMBINATION',...
'position',[350 150 120 65]);

button21 = uicontrol(gcf,...
'style','push',...
'position',[318 230 192 25],...
'string','FEATURE COMBINATION',...
'callback','initfast');

frm5 = uicontrol(gcf,...
'style','frame',...
'position',[318 260 140 85]);

tt5 = uicontrol(gcf,...
'style','text',...
'string','FUZZY C MEANS WITHOUT EVALUATION',...
'position',[332 275 115 65]);

button4 = uicontrol(gcf,...
'style','push',...
'position',[325 265 125 25],...
'string','ALGORITHM',...
'callback','fc_means');

button22 = uicontrol(gcf,...
'style','push',...
'position',[337 125 100 25],...
'string','GENETIC',...
'callback','genetic4');

button23 = uicontrol(gcf,...
'style','push',...
'position',[337 95 100 25],...
```

```
'string','RANDOM',...
'callback','random');

button24 = uicontrol(gcf,...
'style','push',...
'position',[337 65 145 25],...
'string','PSEUDO-EXHAUSTIVE',...
'callback','feature4');
```

****

```
% THIS PROGRAM COMPARES RESULTS BY DIFFERENT SET-UPS
% OF THE 'm'. AN EXAMPLE:

w_comp=zeros(1,669);
n_comp=zeros(1,669);

index=[1 3 5 15 17 19 22 29 30 31 33 36 37 38 39 40 50];
selindex=1:17;

w_comp(index) = selw_percent(selindex) - w_percent(index);
n_comp(index) = seln_percent(selindex) - n_percent(index);

Rindex=[70 141 155 177 197 200 202 211 214 216 235 449 450 453 458 462 600];
selindex=18:34;

w_comp(Rindex) = selw_percent(selindex) - w_percent(Rindex);
n_comp(Rindex) = seln_percent(selindex) - n_percent(Rindex);


%for 11 newis;

newindices=[4 12 18 52 68 82 176 395 451 459 460 ];
w_comp(newindices) = w_percent(newindices);
n_comp(newindices) = n_percent(newindices);

in=[1 3 4 5 12 15 17 18 19 22 29 30 31 33 36 37 38 39 40 50 52 68 70 82 141 155 ...
176 177 197 200 202 211 214 216 235 395 449 450 451 453 458 459 460 462 600];

[in;m2w_percent;m2n_percent;w;w_comp(in) ;n_comp(in)]'
```

****

```
% ANOTHER EXAMPLE:

w_comp=zeros(1,669);
n_comp=zeros(1,669);

index=[1 3 4 5 12 15 17 18 19 22 29 30 31 33 36 37 38 39 40 50 52 68 ...
70 82 141 155 176 177 197 200 211 214 216 235 395 449 450 451];
selindex=1:38;

w_comp(index) = selw_percent(selindex) - w_percent(index);
n_comp(index) = seln_percent(selindex) - n_percent(index);

Rindex=[453 458 459 460 462 600];
selindex=40:45;

w_comp(Rindex) = selw_percent(selindex) - w_percent(Rindex);
n_comp(Rindex) = seln_percent(selindex) - n_percent(Rindex);


%for 1 newy;

newindices=[452];
w_comp(newindices) = w_percent(newindices);
n_comp(newindices) = n_percent(newindices);

in=[1 3 4 5 12 15 17 18 19 22 29 30 31 33 36 37 38 39 40 50 52 68 ...
70 82 141 155 176 177 197 200 211 214 216 235 395 449 450 451 452 ...
453 458 459 460 462 600];

[in;m2w_percent;m2n_percent;w;w_comp(in) ;n_comp(in)]'
```

****

```
% THIS PROGRAM SELECT AND EVALUATE FEATURE GROUPS
% ACCORDING TO THE THRESHOLD.

dimension=669;

l=0;
for g=1:dimension
%-------------------ATTENTION: Change parameters for m=3...

if( (n_percent(g)<=40) & (w_percent(g)<=40) )

l=l+1;
gg(l)=g;
m2wrong_dcps(l)=wrong_dcps(g);
m2w_percent(l)=w_percent(g);

m2w_ok(l)=100-m2w_percent(l);
```

```
m2wrong_nons(l)=wrong_nons(g);
m2n_percent(l)=n_percent(g);

m2n_ok(l)=100-m2n_percent(l);

m2z(l)=z(g);

if( (n_percent(g)<=25) | (w_percent(g)<=25) )
w(l)=1.1111;
else
w(l)=0;
end

end

end
l

fprintf('m2ft_#, m2wrong_dcps, m2w_ok, m2wrong_nons, m2n_ok, m2iterations, bests');
h=1:l;
[gg(h)
m2wrong_dcps(h)
m2w_ok(h)
m2wrong_nons(h)
m2n_ok(h)
m2z(h)
w(h)]'

*********************************************************************
% THIS PROGRAM REPRESENTS ONE THE RANDOM SEARCH
% FOR 4-TUPLE FEATURE COMBINATIONS.

indi=0;
for i=1:10000

aaa = round(10*rand(1,4)); %-----------4*ft--&--size of no=14
%           if aaa(1)>=7, aaa(1)=aaa(1)-5;end;
%           if aaa(2)>=7, aaa(2)=aaa(2)-5;end;
%           if aaa(3)>=7, aaa(3)=aaa(3)-5;end;

if aaa(1)==0, aaa(1)=11;end;
if aaa(2)==0, aaa(2)=12;end;
if aaa(3)==0, aaa(3)=13;end;
if aaa(4)==0, aaa(4)=14;end;

while ( (aaa(1)==aaa(2)) | (aaa(1)==aaa(3)) | (aaa(2)==aaa(3)) ...
| (aaa(2)==aaa(4)) | (aaa(1)==aaa(4)) ...
| (aaa(3)==aaa(4)) )
            aaa = round(10*rand(1,4));
%           if aaa(1)>=7, aaa(1)=aaa(1)-5;end;
%           if aaa(2)>=7, aaa(2)=aaa(2)-5;end;
%           if aaa(3)>=7, aaa(3)=aaa(3)-5;end;
            if aaa(1)==0, aaa(1)=11;end;
            if aaa(2)==0, aaa(2)=12;end;
            if aaa(3)==0, aaa(3)=13;end;
            if aaa(4)==0, aaa(4)=14;end;
end,

i,
indi,
aaa,

clear Xselect;
Xselect=Xsel(aaa,:);

initfast;

%--------------ATTENTION: LIMITATIONS -----------* %if *-----------

%if( ((nn>=80) & (ww>=80)) | ((nn>=84) | (ww>=84)) )
if (((nn>=81)&(ww>=81)) | ((nn>=86)&(ww>=79)))          %-------- && 4*ft x3m5m2
%if ( (nn>=70) & (ww>=80) )                              %--------4*ft x1m5
            indi=indi+1;
            a1_combin(indi) = aaa(1);
            a2_combin(indi) = aaa(2);
            a3_combin(indi) = aaa(3);
            a4_combin(indi) = aaa(4);%-----------4*ft
            n_combres(indi) = nn;
            w_combres(indi) = ww;
            fprintf('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>');
            size(a1_combin)
            fprintf('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>');
end

end

j=1:indi;
[a1_combin(j)
a2_combin(j)
```

```
a3_combin(j)
a4_combin(j)
n_combres(j)
w_combres(j)]'

**********************************************************************

% This program exhaustively tests all possible combinations of
% size eight in x3 from the number of features.  It then records the ones
% that meet the if-then criteria below.
% clear('init') for normal initialization.

load x3

features=[81 111 450 452 197 459 30 ]
n=length(features)

for f=1:n
 Xsel(f,1:100)=x3(features(f),1:100);
end

if exist('init')==1
		% program continuation. No need to initialize other variables.
		i1=init(1);
		i2=init(2);
		i3=init(3);
		i4=init(4);
		i5=init(5);
		i6=init(6);
		i7=init(7);
		i8=init(8);
else
		% initialize all variables.
		indi=0;
		record=[];
		i1=1;
		i2=2;
		i3=3;
		i4=4;
		i5=5;
		i6=6;
		i7=7;
		i8=8;
end


while i1<=n-7
while i2<=n-6
while i3<=n-5
while i4<=n-4
while i5<=n-3
while i6<=n-2
while i7<=n-1
while i8<=n

aaa=[i1 i2 i3 i4 i5 i6 i7 i8]
indi

clear Xselect;
Xselect=Xsel(aaa,:);

initfast;

%--------------ATTENTION: LIMITATIONS ----------" %if "-----------
ave = (nn+ww)/2;

if( ((nn>=81)&(ww>=81)) | (ave>=83) )
		indi=indi+1;
		record=[record ; features(aaa) nn ww];
		fprintf('>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>');
		[features(aaa) nn ww]
end

i8=i8+1;
end		% end i8 loop
i7=i7+1;
i8=i7+1;
end		% end i7 loop
i6=i6+1;
i7=i6+1;
i8=i7+1;
end		% end i6 loop
i5=i5+1;
i6=i5+1;
i7=i6+1;
i8=i7+1;
end		% end i5 loop
i4=i4+1;
i5=i4+1;
i6=i5+1;
i7=i6+1;
```

```
i8=i7+1;
end          % end i4 loop
i3=i3+1;
i4=i3+1;
i5=i4+1;
i6=i5+1;
i7=i6+1;
i8=i7+1;
end          % end i3 loop
i2=i2+1;
i3=i2+1;
i4=i3+1;
i5=i4+1;
i6=i5+1;
i7=i6+1;
i8=i7+1;
end          % end i2 loop
i1=i1+1;
i2=i1+1;
i3=i2+1;
i4=i3+1;
i5=i4+1;
i6=i5+1;
i7=i6+1;
i8=i7+1;
end          % end i1 loop

record
```

****

```
% Genetic algorithm in search of the optimal n-tuple
% from a gene pool of features.
% This version records the actual feature numbers in the
% matrix 'record', not the index!!
% x3. Set m in initfast.
% set init=1 for automatic initialization

comment='x3, m=5, 15-tuple.'
n=15;
load x3
clear Xselect;
features=[9 11 30 50 39 81 235 358 359 363 449 197 29 450 453 457 458 478 ...
          111 452 482 361 15 36 37 32 8 67 79 460]

% features=[4 5 8 9 12 18 19 22 29 30 33 36 39 40 50 56 62 76 79 81 ...
%           111 114 163 197 235 358 359 361 363 403 449 450 452 453 456 457 ...
%           458 477 478 482 534 625 ]

feature_num=length(features)

for f=1:feature_num
 Xsel(f,1:100)=x3(features(f),1:100);
end
clear x3;
clear average_fitness;

if init==1
        % initialize population size, crossover rate, mutation rate, etc.
        population_size=200;
        mutation_rate=0.001;
        crossover_rate=0.7;
        record=zeros(20,n+3);
        indi=0;

        % initialize population
        rand('uniform');
        population=fix((feature_num - .0000001) .* rand(population_size,n)) + 1;
end

% start evolution
for generation=1:100000
generation

% test the population for fitness
for f= 1:population_size
Xselect = Xsel(population(f,:),:);%
initfast;                               % test each individual

fitness(f) = abs((nn+ww)/2 - 20);       % subtract 20 to exaggerate the
                                        % difference in fitness ratio
        %if ( ((nn>=70)&(ww>=70)) | (fitness>=56) | ((nn<=20)&(ww<=20)) )
        if (  (fitness(f)>=65)  |  ((nn<=20)&(ww<=20))  )
                indi=indi+1
                record(indi,:) = [features(population(f,:)) generation nn ww];%
                [features(population(f,:)) generation nn ww]
        end
end

% display average fitness in percentage
average_fitness(generation)=mean(fitness) + 20
```

```
% REPRODUCTION !!
% reorder the fitness values for easier computation
fit_measure(1)=fitness(1);
for f=2:population_size
fit_measure(f)=fit_measure(f-1)+fitness(f);
end

for f=1:population_size
% randomly pick one individual to copy into the new population
% individuals with higher fitness values are more likely to survive
temp=fit_measure(population_size) .* rand;
index=find(abs(fit_measure-temp) == min(abs(fit_measure-temp)));
        if temp <= fit_measure(index(1))
                new_population(f,:)=population(index(1),:);%
        else
                new_population(f,:)=population(index(1)+1,:);%
        end
end
population=new_population;

% CROSSOVER !!
f=1;
while f <= population_size
        if rand <= crossover_rate
                mate = f;
                crossover = 0;
                while (f < population_size) & (crossover==0)
                        f=f+1;
                        if rand <= crossover_rate
                          % actual crossover
                          crossover = 1;
                          temp=fix((n - 1.00001) .* rand) + 2;%
                          gene_temp=population(mate,temp:n);%
                          population(mate,temp:n)=population(f,temp:n);%
                          population(f,temp:n)=gene_temp;
                        end
                end
        end
        f=f+1;
end

% MUTATION !!
% Note: Modified Aug. 19 due to a bug
num_mutation=population_size .* mutation_rate .* n .* (randn + 1);
for f=1:num_mutation
 population(fix((population_size-0.000001).*rand+1),fix((n-0.00001)*rand+1))...
        = fix((feature_num - 0.000001) * rand + 1);
end

% save record in case of crashing
save crashrec comment record average_fitness

% go to next generation
end

% display record of good individuals in history
comment
record

% [sort(record(1:indi,1:n))' record(1:indi,n+1:n+3)]

************************************************************************
% SELECTION AND INITIALIZATION OF THE DATA CENTERS
% FOR THE LMS FILTER.

% "initrain_sess" = Polygraph sessions which are used for
% INITialization of the "data_centers" and TRAINing.

% The "initrain" sessions are set in a way that the 1st part
% (before the "border") represents the non-decptive and the
% 2nd part (after the border) the deceptive sessions.

clear;
%*** To be set for each polydat_i (ftx3, ftx2, ftx1): ******************
%*                                                                    *
        whichfeatures_3 = [1:30];
        nondsessions_3 = [11:50];
                %[1 6 8 9 12 16 18 21 24 27 28 32 35 44 48];
        dcpsessions_3 = [51:90];
                %[51 53 58 59 63 67 72 75 82 85 88 89 93 95 100];
%*                                                                    *
        whichfeatures_2 = [];
        nondsessions_2 = [];
        dcpsessions_2 = [];
%*                                                                    *
        whichfeatures_1 = [];
        nondsessions_1 = [];
        dcpsessions_1 = [];
%*                                                                    *
```

```
%* ATTENTION: The DIMENSION of each "whichfeatures_..." is to be equal!     *
%* (or zero)                                                                 *
%****************************************************************************

if length(whichfeatures_3) ~= length(whichfeatures_2) | ...
  length(whichfeatures_2) ~= length(whichfeatures_1) ,

        fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
        fprintf('Check "whichfeatures"! They are different big!\n');
        fprintf('The dimensions are as following:\n');
        fprintf('\n');
        fprintf('  1st  2nd  3rd\n');
        disp( [length(whichfeatures_1), length(whichfeatures_2), ...
        length(whichfeatures_3)] )
        fprintf('\n');
        fprintf('YOU DO NOT NEED TO CHANGE THE EMPTY ONES!\n');
        fprintf('IF THAT'S THE CASE: PRESS ANY KEY TO CONTINUE.\n');
        fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
        pause;

end;

border = length(nondsessions_3) + length(nondsessions_2) ...
    + length(nondsessions_1);

%%%% polydat_3:

if size(nondsessions_3,1) ~= 0,

        load c:\users\ramin\fcm\multidim\ftx3;

        dim = length(whichfeatures_3);
        f=1:dim;
        Ntemp_3(f,:) = x3(whichfeatures_3(f), nondsessions_3);
        Dtemp_3(f,:) = x3(whichfeatures_3(f), dcpsessions_3);

        clear x3;
end;

%%%% polydat_2

if size(nondsessions_2,1) ~= 0,

        load c:\users\ramin\fcm\multidim\ftx2;

        dim = length(whichfeatures_2);
        f=1:dim;
        Ntemp_2(f,:) = x2(whichfeatures_2(f), nondsessions_2);
        Dtemp_2(f,:) = x2(whichfeatures_2(f), dcpsessions_2);

        clear x2;
end;

%%%% polydat_1

if size(nondsessions_1,1) ~= 0,

        load c:\users\ramin\fcm\multidim\ftx1;

        dim = length(whichfeatures_1);
        f=1:dim;
        Ntemp_1(f,:) = x1(whichfeatures_1(f), nondsessions_1);
        Dtemp_1(f,:) = x1(whichfeatures_1(f), dcpsessions_1);

        clear x1;
end;

initrain_sess = [Ntemp_3'; Ntemp_2'; Ntemp_1'; ...
                        Dtemp_3'; Dtemp_2'; Dtemp_1'];

howmany = size(initrain_sess,1);


mesh(initrain_sess);


% TWO FEATURES AT A TIME - PLOT EXAMPLE:
%plot(initrain_sess(1:40,1),initrain_sess(1:40,4),'.y')
%hold on
%plot(initrain_sess(41:80,1),initrain_sess(41:80,4),'.r')
****************************************************************************

% SELECTION AND INITIALIZATION FOR LMS FILTER.

% The "initrain" data represents Polygraph sessions which are used for
% INITialization and TRAINing of the "data_centers" and input data.
```

```
% The "initrain" data are set in a way that the 1st part - before the
% "(TC_)border" - represents the non-decptive and the second part
% - after the "(TC_)border" - the deceptive sessions.

% The prefix "nond" represents the non_decptive, and "dcp" the deceptive
% elements.

clear;
%*********************************************************************

%********** TO BE SET FOR EACH polydat_i (ftx3, ftx2, ftx1): *******
%*                                                                *
%* First for the data_centers:                                    *
%*                                                                *
            nondsessions_3 = [1:20];
                        %[1 6 8 9 12 16 18 21 24 27 28 32 35 44 48];
            dcpsessions_3 = [51:70];
                        %[51 53 58 59 63 67 72 75 82 85 88 89 93 95 100];
%*                                                                *
            nondsessions_2 = [];
            dcpsessions_2 = [];
%*                                                                *
            nondsessions_1 = [];
            dcpsessions_1 = [];
%*                                                                *
%*                                                                *
%* Now for the input data for which the filter is to be (T)rained *
%* to (C)lassify:                                                 *
%*                                                                *
            TC_nondsessions_3 = [1:30];
            TC_dcpsessions_3 = [51:80];                           *
                                                                  *
            TC_nondsessions_2 = [];
            TC_dcpsessions_2 = [];
%*                                                                *
            TC_nondsessions_1 = [];
            TC_dcpsessions_1 = [];
%*                                                                *
%*                                                                *
%* And finally for the selected features:
%*                                                                *
            whichfeatures_3 = [1:30];
            whichfeatures_2 = [];
            whichfeatures_1 = [];
%*                                                                *
%* ATTENTION: The DIMENSION of each "whichfeatures_..." is to be equal! *
%* (or zero)                                                      *
%*******************************************************

if length(whichfeatures_3) ~= length(whichfeatures_2) | ...
  length(whichfeatures_2) ~= length(whichfeatures_1) ,

            fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
            fprintf('Check "whichfeatures"! They are different big!\n');
            fprintf('The dimensions are as following:\n');
            fprintf('\n');
            fprintf('  1st  2nd  3rd\n');
            disp( [length(whichfeatures_1), length(whichfeatures_2), ...
            length(whichfeatures_3)] )
            fprintf('\n');
            fprintf('YOU DO NOT NEED TO CHANGE THE EMPTY ONES!\n');
            fprintf('IF THAT'S THE CASE: PRESS ANY KEY TO CONTINUE.\n');
            fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
            pause;

end;

border = length(nondsessions_3) + length(nondsessions_2) ...
    + length(nondsessions_1);

TC_border = length(TC_nondsessions_3) + length(TC_nondsessions_2) ...
            + length(TC_nondsessions_1);


%%% polydat_3:

dim = length(whichfeatures_3);
if dim ~= 0,

            load c:\users\ramin\fcm\multidim\ftx3;
            f=1:dim;

            if length(TC_nondsessions_3) ~= 0,
            TC_Ntemp_3(f,:) = x3(whichfeatures_3(f), TC_nondsessions_3);
            end;

            if length(TC_dcpsessions_3) ~= 0,
            TC_Dtemp_3(f,:) = x3(whichfeatures_3(f), TC_dcpsessions_3);
            end;
```

```
        if length(nondsessions_3) ~= 0,
        Ntemp_3(f,:) = x3(whichfeatures_3(f), nondsessions_3);
        end;

        if length(dcpsessions_3) ~= 0,
        Dtemp_3(f,:) = x3(whichfeatures_3(f), dcpsessions_3);
        end;

        clear x3;
end;

%%% polydat_2

dim = length(whichfeatures_2);
if dim ~= 0,

        load c:\users\ramin\fcm\multidim\ftx2;
        f=1:dim;

        if length(TC_nondsessions_2) ~= 0,
        TC_Ntemp_2(f,:) = x2(whichfeatures_2(f), TC_nondsessions_2);
        end;

        if length(TC_dcpsessions_2) ~= 0,
        TC_Dtemp_2(f,:) = x2(whichfeatures_2(f), TC_dcpsessions_2);
        end;

        if length(nondsessions_2) ~= 0,
        Ntemp_2(f,:) = x2(whichfeatures_2(f), nondsessions_2);
        end;

        if length(dcpsessions_2) ~= 0,
        Dtemp_2(f,:) = x2(whichfeatures_2(f), dcpsessions_2);
        end;

        clear x2;
end;

%%% polydat_1

dim = length(whichfeatures_1);
if dim ~= 0,

        load c:\users\ramin\fcm\multidim\ftx1;
        f=1:dim;

        if length(TC_nondsessions_1) ~= 0,
        TC_Ntemp_1(f,:) = x1(whichfeatures_1(f), TC_nondsessions_1);
        end;

        if length(TC_dcpsessions_1) ~= 0,
        TC_Dtemp_1(f,:) = x1(whichfeatures_1(f), TC_dcpsessions_1);
        end;

        if length(nondsessions_1) ~= 0,
        Ntemp_1(f,:) = x1(whichfeatures_1(f), nondsessions_1);
        end;

        if length(dcpsessions_1) ~= 0,
        Dtemp_1(f,:) = x1(whichfeatures_1(f), dcpsessions_1);
        end;

        clear x1;
end;

TC_initrain = [TC_Ntemp_3'; TC_Ntemp_2'; TC_Ntemp_1'; ...
              TC_Dtemp_3'; TC_Dtemp_2'; TC_Dtemp_1'];

cent_initrain = [Ntemp_3'; Ntemp_2'; Ntemp_1'; ...
                Dtemp_3'; Dtemp_2'; Dtemp_1'];

**********************************************************************
% LMS FUZZY ADAPTIVE FILTER.

function [new_theta, new_data_centers, new_sigma, output_label] = ...
adaptzzy(theta, data_centers, sigma, input_vect, desire, step)
%fprintf('size(theta):');size(theta),
%fprintf('size(sigma):');size(sigma),

% Get the dimensions of matrices and verify their consistency:
[label_no, ft_no] = size(data_centers);
if ([label_no, ft_no] ~= size(sigma)) | ([1, ft_no] ~=size(input_vect)) |...
  ([label_no, 1] ~= size(theta))

        error('matrix dimensions are wrong! ')
end;
%+++
```

```
% Evaluate Gaussian membershipfunctions:

distances = (ones(label_no,1) * input_vect) - data_centers;
%fprintf('size(distances):');size(distances),
% To creat compatible dimensions: Fill input_vect down into an
% (label_no x ft_no) matrix, so that it is the same input for all
% (label_no) rules, and then subtract data_centers from it.

a = exp(  -0.5 .* sum( ((distances ./ sigma) .^2 )' )'  );
% Without "sum":  a = Uik i.e. membership values
% etc.etc...(conventional way)
%+++
%fprintf('size(a):');size(a),

% Centroidal defuzzification:
b = sum(a);%fprintf('size(b):');size(b),
output_label = sum(theta .* a) / b;
%+++


% Adaption:

temp1 = step .* (desire - output_label) .* a ./b;
new_theta = theta + temp1;

temp2 = ( (temp1 .* (theta - output_label) ) * ones(1, ft_no)) .* ...
              distances ./ (sigma .^2);
new_data_centers = data_centers + temp2;

new_sigma = sigma + temp2 .* distances ./ sigma;
%+++


return;
**********************************************************************
% LMS FILTER INTIALIZATION (TRAINING AND TESTING)
% FIRST VERSION

% clear everything!
clear;

%loading ...:
load c:\users\ramin\fcm\multidim\ftx3;

which_features = 1:100;%------------------------to change!!!


% the data from the 'person' who is to be tested:
person = 2;
testperson = x3(which_features,person)';


polysession(1,:) = x3(which_features,1)';%nondecp
%%%[x3(81,1), x3(111,1), x3(235,1), x3(450,1), x3(452,1)];

polysession(16,:) = x3(which_features,100)';%decp
%%%[x3(81,100), x3(111,100), x3(235,100), x3(450,100),...
%%%x3(452,100)];              % polygraph data for two sessions,
                              % i.e one truthful & one decpetive

polysession(2,:) = x3(which_features,48)';%nondecp
polysession(3,:) = x3(which_features,5)';%nondecp
polysession(4,:) = x3(which_features,8)';%nondecp
polysession(5,:) = x3(which_features,9)';%nondecp
polysession(6,:) = x3(which_features,12)';%nondecp
polysession(7,:) = x3(which_features,16)';%nondecp
polysession(8,:) = x3(which_features,18)';%nondecp
polysession(9,:) = x3(which_features,21)';%nondecp
polysession(10,:) = x3(which_features,24)';%nondecp
polysession(11,:) = x3(which_features,27)';%nondecp
polysession(12,:) = x3(which_features,28)';%nondecp
polysession(13,:) = x3(which_features,32)';%nondecp
polysession(14,:) = x3(which_features,35)';%nondecp
polysession(15,:) = x3(which_features,44)';%nondecp

polysession(17,:) = x3(which_features,95)';%decp
polysession(18,:) = x3(which_features,93)';%decp
polysession(19,:) = x3(which_features,89)';%decp
polysession(20,:) = x3(which_features,88)';%decp
polysession(21,:) = x3(which_features,85)';%decp
polysession(22,:) = x3(which_features,82)';%decp
polysession(23,:) = x3(which_features,75)';%decp
polysession(24,:) = x3(which_features,72)';%decp
polysession(25,:) = x3(which_features,67)';%decp
polysession(26,:) = x3(which_features,63)';%decp
polysession(27,:) = x3(which_features,59)';%decp
polysession(28,:) = x3(which_features,58)';%decp
```

```
polysession(29,:) = x3(which_features,53)';%decp
polysession(30,:) = x3(which_features,51)';%decp
[howmany, dim] = size(polysession),% "howmany" must be even!
half = howmany/2;

clear x3;       %save memory & clear

%+++

%initialiation & clear:

step = 0.005;
output = zeros(1, 2)


output_mean = [1, 2]'

input_mean = polysession;
input_width = 1 * ones(howmany, dim);

% Testing(see 100 for des)

        [dummy, dummy, dummy, output] = ...
        adaptzzy(output_mean, input_mean, input_width, testperson,...
        100, step);
                                        % Test how good the output is at
                                        % the beginning.


end,
output
pause;

figure(1);clg
plot(output,'.');
%plot(output_mean,'.b');
hold on;
%mesh(input_width);
**********************************************************************
% SEE ABOVE - SECOND VERSION.
%User interface to improve!

% INITIALIZATION:
% +++++++++++++++

step = 0.5;                     % Learning factor

% The prefix "TC" represents the input data for which the filter
% is to be (T)rained to (C)lassify:

TC_howmany = size(TC_initrain, 1);
[howmany, dim] = size(cent_initrain);     % representing data_centers

clear output;
output = zeros([TC_howmany, 1]);

% "+1" represents the nondeceptive and "-1" the deceptive data:
init_theta_non = +1 * ones(border, 1);
init_theta_dcp = -1 * ones((howmany-border), 1);

output_mean = [init_theta_non; init_theta_dcp];        % ~ data_centers

input_mean = cent_initrain;
input_width = 100 * ones(howmany, dim);

% +++++++++++++++


% Before any training ...
% Test how good the output is at the beginning:

for k=1:TC_howmany

        if k<=TC_border
        des=+1;
        else
        des=-1;
        end

        [dummy, dummy, dummy, output(k)] = ...
        adaptzzy(output_mean, input_mean, ...
        input_width, TC_initrain(k,:),...
        des, step);

end,
clear dummy;
output,

figure(1);clg
plot(output,'+');
```

```
%plot(output_mean,'*b');
hold on;
pause;
%mesh(input_width);

% Starting training:  DO A BETTER USERINTERFACE!
for i=1:30
for j=1:5
        for k=1:TC_howmany

                if k<=TC_border
                des=+1;
                else
                des=-1;
                end

                [output_mean, input_mean, input_width, output(k)] = ...
                adaptzzy(output_mean, input_mean, input_width, ...
                TC_initrain(k,:), des, step);
        end,
end,
output,

figure(1);
plot(output,'.');           %axis([1 100 -0.2 2.1]);
%plot(output_mean,'*b');
%mesh(input_width);
%pause;

end;


%************ SAVING THE FILTER CHARACTERISTICS: ******************

fprintf('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n');
fprintf('IF YOU WANT TO SAVE THE CHARACRERISTICS OF THIS FILTER,\n');
fprintf('PLEASE TYPE ANY NUMBER(#) FROM 1-99!\n');
fprintf('THIS FILTER WILL BE THEN SAVED AS "filter#" !\n');

clear numb;
numb = input('The filter number(#) is:');
% By default: numb=[], i.e. nothing will be saved.

if numb ~=[],
        numb = int2str(numb);
        com =['save ', 'filter', numb, ...
                ' whichfeatures_3', ...
                ' whichfeatures_2', ...
                ' whichfeatures_1', ...
                ' output_mean', ' output_mean', ...
                ' input_mean', ' input_width'];
        eval(com);
end;
*********************************************************************
% CREATING THE ELLIPTICAL CLUSTERS FOR THE VISUAL
% INSPECTIONS - AND ALSO FOR STTING THE RULES.

function [x,y]=ellipse(xcenter,ycenter,xwidth,ywidth)
angle=[0:0.02*pi:2*pi];
x=xwidth .* cos(angle) + xcenter;
y=ywidth .* sin(angle) + ycenter;
plot(x,y,'-')


*********************************************************************
% TEMPORARY LMS SETTING - TEST

function output_label=fuzztemp(input_vect)
theta=[ 1 1 -1 -1]';
data_centers=[ -1 -0.5 ; 0 -0.25 ; 0 0 ; 1 0.3 ];
sigma=[ 0.5 0.8 ; 0.5 0.25 ; 0.1 0.2 ; 0.6 0.5 ];


% Get the dimensions of matrices and verify their consistency:
[label_no, ft_no] = size(data_centers);
if ([label_no, ft_no] ~= size(sigma)) | ([1, ft_no] ~=size(input_vect)) |...
  ([label_no, 1] ~= size(theta))

        error('matrix dimensions are wrong! ')
end;
%+++


% Evaluate Gaussian membershipfunctions:

distances = (ones(label_no,1) * input_vect) - data_centers;

a = exp(  -0.5 .* sum( ( (distances ./ sigma) .^2 )' )'  );
```

```
% Centroidal defuzzification:
b = sum(a);
output_label = sum(theta .* a) / b;
output_label = output_label .^ 2;

return;

*********************************************************************

% LMS FILTER TESTING.
% Experimenting with the use of adaptive fuzzy logic
% in polygraph classification.

init=input('Do you want to initialize all parameters? ','s');
if init=='y'
 % Initialize the parameters for fuzzy LMS algorithm.
 % Output of 1 means nondeceptive
 % Output of -1 means deceptive
 % length(output_mean) = # of rules
 fprintf('initializing\n');
 output_mean=[ 1 1 -1 -1]';
 % input_mean=[ centers of first rule ; centers of second rule ; etc. ];
 input_mean=[ -1 -0.5 ; 0 -0.1 ; 0 0 ; 1 0.3 ];
 % input_width=[ widths of first rule ; widths of second rule ; etc. ];
 input_width=[ 0.5 1.3 ; 0.5 0.25 ; 0.1 0.2 ; 0.6 0.5 ];

 features=[451 452];          % Select the features
 step=0.005;                  % Select learning rate

 % Select training data
 ndcp_3=1:15;                 % Nondeceptive sessions in x3 for training
 dcp_3=51:65;                 % Deceptive sessions in x3 for training
 ndcp_2=[];
 dcp_2=[];
 ndcp_1=[];
 dcp_1=[];                    % Note that nondeceptive data in x1, x2, and x3
                              % are the same, so ndcp_2 and ndcp_1 are really
                              % redundant.

 load x3;
 load x2;
 load x1;
 Ntrain=[x1(features,ndcp_1) x2(features,ndcp_2) x3(features,ndcp_3)]';
 Dtrain=[x1(features,dcp_1) x2(features,dcp_2) x3(features,dcp_3)]';

 % Select testing data
 ndcp_3=[];                   % Nondeceptive sessions in x3 for testing
 dcp_3=66:100;
 ndcp_2=[];
 dcp_2=[51:100];
 ndcp_1=16:50;
 dcp_1=[51:100];              % Note that nondeceptive data in x1, x2, and x3
                              % are the same, so ndcp_2 and ndcp_1 are really
                              % redundant.

 Ntest=[x1(features,ndcp_1) x2(features,ndcp_2) x3(features,ndcp_3)]';
 Dtest=x1(features,dcp_1)';
 Dtest2=x2(features,dcp_2)';
 Dtest3=x3(features,dcp_3)';
 clear x1;
 clear x2;
 clear x3;
 clear record;
 epoch=0;
end

 % Test fuzzy system before any training
 % Test training data first
 clear Noutput;
 clear Doutput;
 [Ntr,dummy]=size(Ntrain);    % Ntr = total # of nondeceptive sessions
 [Dtr,dummy]=size(Dtrain);    % Dtr = total # of deceptive sessions
 if Ntr ~= Dtr
         error('Number of nondeceptive and deceptive training data mismatch');
 end
 for i=1:Ntr
         [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                 input_width,Ntrain(i,:),1,step);
         [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                 input_width,Dtrain(i,:),-1,step);
 end
 % Record results
 record(epoch+1,1:2)=[(length(find(Noutput>0))/Ntr) (length(find(Doutput<0))/Dtr) ];
 squared_error(epoch+1,1:2)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
 fprintf('percent correct nondeceptive and deceptive detections for training data:\n');
 disp(record(epoch+1,1:2))


 % Now test testing data
 clear Noutput;
 clear Doutput;
 [Nte,dummy]=size(Ntest);     % Nte = total # of nondeceptive sessions
```

```
for i=1:Nte
        [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Ntest(i,:),1,step);
end
[Dte,dummy]=size(Dtest);    % Dte = total # of deceptive sessions in x1
for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest(i,:),-1,step);
end
squared_error(epoch+1,3:4)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
record(epoch+1,3:4)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte)];
[Dte,dummy]=size(Dtest2);   % Dte = total # of deceptive sessions in x2
clear Doutput;
for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest2(i,:),-1,step);
end
squared_error(epoch+1,5:6)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
record(epoch+1,5:6)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte)];
[Dte,dummy]=size(Dtest3);   % Dte = total # of deceptive sessions in x3
clear Doutput;
for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest3(i,:),-1,step);
end
squared_error(epoch+1,7:8)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
record(epoch+1,7:8)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte)];

 fprintf('training,x1,x2,x3:\n');
 disp(record(epoch+1,:))


 % Start training and testing
 fprintf('results after training\n')
 while epoch<100000
 epoch=epoch+1
 clear Noutput;
 clear Doutput;
  % Training
  for i=1:Ntr
        [output_mean,input_mean,input_width,Noutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Ntrain(i,:),1,step);
        [output_mean,input_mean,input_width,Doutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Dtrain(i,:),-1,step);
  end
  % end one epoch
 % Test training data
  for i=1:Ntr
        [dummy,dummy,dummy,Noutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Ntrain(i,:),1,step);
        [dummy,dummy,dummy,Doutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Dtrain(i,:),-1,step);
  end
  % Record results of training data at the end of an epoch
  squared_error(epoch+1,1:2)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
  record(epoch+1,1:2)=[(length(find(Noutput>0))/Ntr) (length(find(Doutput<0))/Dtr) ];


  % Now test testing data
  clear Noutput;
  clear Doutput;
  [Nte,dummy]=size(Ntest);
  for i=1:Nte
        [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Ntest(i,:),1,step);
  end
  [Dte,dummy]=size(Dtest);
  for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest(i,:),-1,step);
  end
  squared_error(epoch+1,3:4)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
  record(epoch+1,3:4)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte) ];
  [Dte,dummy]=size(Dtest2);   % Dte = total # of deceptive sessions in x2
  clear Doutput;
  for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest2(i,:),-1,step);
  end
  squared_error(epoch+1,5:6)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
  record(epoch+1,5:6)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte)];
  [Dte,dummy]=size(Dtest3);   % Dte = total # of deceptive sessions in x3
  clear Doutput;
  for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest3(i,:),-1,step);
```

```
end
squared_error(epoch+1,7:8)=[mean((1-Noutput).^2) mean((Doutput+1).^2)];
record(epoch+1,7:8)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte)];

 fprintf('training,x1,x2,x3:\n');
 disp(record(epoch+1,:))

end          % Go to next epoch

**********************************************************************

% Experimenting with the use of adaptive fuzzy logic
% in polygraph classification.

for trial=1:1
 % Initialize the parameters for fuzzy LMS algorithm.
 % Output of 1 means nondeceptive
 % Output of -1 means deceptive
 % length(output_mean) = # of rules
 fprintf('initializing\n');
 output_mean=[ 1 1 -1 -1]';
 % input_mean=[ centers of first rule ; centers of second rule ; etc. ];
 input_mean=[ -1 -0.5 ; 0 -0.25 ; 0 0 ; 1 0.3 ];
 % input_width=[ widths of first rule ; widths of second rule ; etc. ];
 input_width=[ 0.5 0.8 ; 0.5 0.25 ; 0.1 0.2 ; 0.6 0.5 ];

 features=[451 452];          % Select the features
 step=0.005;                  % Select learning rate
 trainers=10;                 % Select # of training samples from each category

 % Select training data
 temp_n=randperm(50);
 temp_d=50+randperm(50);
 ndcp_3=[1:5 7:10 12 13 15 16 18:20 22 23 25 26 28 29 31 32 34 35 37 38 40 41 43 44 46:49];
 dcp_3=[51 54 57 60 64 67 70 73 76 79 82 85];% Deceptive sessions in x3 for training
 ndcp_2=[];
 dcp_2=[51 53 56 59 62 65 68 71 74 78 81 84];
 ndcp_1=[];
 dcp_1=[51 54 57 59 62 65 68 71 74 77 80 83];
                              % Note that nondeceptive data in x1, x2, and x3
                              % are the same, so ndcp_2 and ndcp_1 are really
                              % redundant.
 load x3;
 load x2;
 load x1;
 Ntrain=[x1(features,ndcp_1) x2(features,ndcp_2) x3(features,ndcp_3)]';
 Dtrain=[x1(features,dcp_1) x2(features,dcp_2) x3(features,dcp_3)]';

 % Select testing data
 ndcp_3=[6 11 14 17 21 24 27 30 33 36 39 42 45 50];
 dcp_3=[52 53 55 56 58 59 61:63 65 66 68 69 71 72 74 75 77 78 80 81 83 84 86:100];
 ndcp_2=[];
 dcp_2=[52 54 55 57 58 60 61 63 64 66 67 69 70 72 73 75:77 79 80 82 83 85:100];
 ndcp_1=[];
 dcp_1=[52 53 55 56 58 60 61 63 64 66 67 69 70 72 73 75 76 78 79 81 82 84:100];
                              % Note that nondeceptive data in x1, x2, and x3
                              % are the same, so ndcp_2 and ndcp_1 are really
                              % redundant.

 Ntest=[x1(features,ndcp_1) x2(features,ndcp_2) x3(features,ndcp_3)]';
 Dtest=[x1(features,dcp_1) x2(features,dcp_2) x3(features,dcp_3)]';
 clear x1;
 clear x2;
 clear x3;
 clear record;
 clear temp_n;
 clear temp_d;
 epoch=0;


 % Test fuzzy system before any training
 % Test training data first
 clear Noutput;
 clear Doutput;
 [Ntr,dummy]=size(Ntrain);    % Ntr = total # of nondeceptive sessions
 [Dtr,dummy]=size(Dtrain);    % Dtr = total # of deceptive sessions
 if Ntr ~= Dtr
          error('Number of nondeceptive and deceptive training data mismatch');
 end
 for i=1:Ntr
          [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                    input_width,Ntrain(i,:),1,step);
          [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                    input_width,Dtrain(i,:),-1,step);
 end
 %% fprintf('Results of training data before training\n');
 %% Noutput
 %% Doutput
 % Record results
 record(epoch+1,1:2)=[(length(find(Noutput>0))/Ntr) (length(find(Doutput<0))/Dtr) ];
 fprintf('percent correct nondeceptive and deceptive detections for training data:\n');
```

```
disp(record(epoch+1,1:2))

% Now test testing data
clear Noutput;
clear Doutput;
[Nte,dummy]=size(Ntest);    % Nte = total # of nondeceptive sessions
for i=1:Nte
        [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Ntest(i,:),1,step);
end
[Dte,dummy]=size(Dtest);    % Dte = total # of deceptive sessions
for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest(i,:),-1,step);
end
if (Nte ~= 0) & (Dte ~= 0)
 %% fprintf('Results of testing data before training\n');
 %% Noutput
 %% Doutput
 % Record results
 record(epoch+1,3:4)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte) ];
 fprintf('percent correct nondeceptive and deceptive detections for testing data:\n');
 disp(record(epoch+1,3:4))
end


% Start training and testing
fprintf('results after training\n')
while epoch<50
epoch=epoch+1
clear Noutput;
clear Doutput;

 % Training
 for i=1:Ntr
        [output_mean,input_mean,input_width,Noutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Ntrain(i,:),1,step);
        [output_mean,input_mean,input_width,Doutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Dtrain(i,:),-1,step);
 end
 % end one epoch
% Test training data
 for i=1:Ntr
        [dummy,dummy,dummy,Noutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Ntrain(i,:),1,step);
        [dummy,dummy,dummy,Doutput(i)]=...
                adaptzzy(output_mean,input_mean,input_width,...
                Dtrain(i,:),-1,step);
 end
%% fprintf('results of training data\n')
%% Noutput
%% Doutput
% Record results of training data at the end of an epoch
record(epoch+1,1:2)=[(length(find(Noutput>0))/Ntr) (length(find(Doutput<0))/Dtr) ];
fprintf('percent correct nondeceptive and deceptive detections for training data:\n')
disp(record(epoch+1,1:2))

if (Nte ~= 0) & (Dte ~= 0)
 % Now test testing data
 clear Noutput;
 clear Doutput;
 for i=1:Nte
        [dummy,dummy,dummy,Noutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Ntest(i,:),1,step);
 end
 for i=1:Dte
        [dummy,dummy,dummy,Doutput(i)]=adaptzzy(output_mean,input_mean,...
                input_width,Dtest(i,:),-1,step);
 end
 %% fprintf('results of testing data\n')
 %% Noutput
 %% Doutput
 record(epoch+1,3:4)=[(length(find(Noutput>0))/Nte) (length(find(Doutput<0))/Dte) ];
 fprintf('percent correct nondeceptive and deceptive detections for testing data:\n')
 disp(record(epoch+1,3:4))
 end
 end           % Go to next epoch
 maximum(trial)=max(record(:,3)+record(:,4));
 temp=[find((record(:,3)+record(:,4))==maximum(trial))' 0 0 0 0 0 ];
 maxima(trial,1:5)=temp(1:5);
 maxima(trial,1:5)
 maximum/2
 end           % Go to next trial
 maximum=maximum/2

 ******************************************************************
```

EPILOGUE - Motivation, challenges and risks

I was easily fascinated by the idea of a lie-detector at the very first moment I heard about it. I thought, 'we are not supposed to lie anyway and a lie-detector can help us find and prevent a major part of the crimes committed in our society. I became even more motivated to do this research by an innovative way of pattern recognition, namely the fuzzy approach.

*But very soon, I also began to realize its danger - while juggling with **numerical data** and being far from the reality of testing actual **human beings** and judging them by an algorithm.*

*An example: **Too 'good' detection rates!***
In my project, I obtained in certain cases up to 97% correct detection rate. That is, indeed, an impressive number. However, the emphasis lies on "certain cases" - not only in this thesis.
A non-technically oriented user of such a product is tempted to put too much trust into these kinds of high rates. Even if we have a stable lie-detector with 99%(!) correct detection, this still means that one out of 100 persons will be judged incorrectly.

*In our daily life, we do not have the natural skill to "see" who is deceptive, but some biological and psychological features that enable us to estimate whether and to what degree someone is lying. This is exactly what I have exploited in this project. In fact, even the fuzzy approach is similar to the human way of categorizing someone's deceptiveness in **soft** terms like "She lies **seldom**" or "He is **often** deceptive", instead of **hard** labeling like "She **is** truthful" or "He **is** deceptive".*

After all, I am convinced that no lie-detector - even if it could work easily with different polygraph formats, and is perfect in technical terms - can ever be constructed with such a high detection rate[63] that one could judge a person without any witnesses or other additional inquiries. We may only use a lie-detector as a helpful "objective" tool, but never as an ultimate decision maker.

My initial goal was to be aware of this responsibilty and not to lose the global perspective while dealing with technical details. I hope I have accomplished this.

I also hope for an environment where we do not judge people who hurt us, but do forgive them. In that case, we ourselves are forgiven too, since all of us deserve to be judged, don't we!

Ramin Djamschidi
San Jose, September 1994.

[63] See e.g. chapter 4.3. for "Outlier effect" and "Performance limitations".

REFERENCES

[Bezdek1981] Bezdek, James C., *Pattern Recognition with Fuzzy Objective Function Algorithm*. Plenum Press, New York and London.

[Bezdek1986] Bezdek, James C. and Siew, K. Chuah, *Generalized K-Nearest Neighbor Rules*, Fuzzy Sets and Systems vol. 18.

[Bezdek1992] Bezdek, James C. and Pal, Sankar K., *Fuzzy Models for Pattern Recognition*, Methods That Search for Structuresin data. IEEE Press, Piscataway, NJ.

[Bezdek1993] Bezdek, James C., *A Review of Probabilstic, Fuzzy, and Neural Models for Pattern Recognition*, Journal of Intelligent and Fuzzy Systems, John Wiley & Sons. Inc.

[Dastmalchi1993] Dastmalchi, Mitra, *Feature Analysis of the Polygraph*. Master's Project, Dept. of Elect. Engr., San Jose State University, California.

[Duda1973] Duda, Richard O. and Hart, Peter E., *Pattern Classification and Scene Analysis*, New York, NY, Wiley.

[Dunn1974] Dunn, J. C., *A fuzzy relative of the ISODATA process and its use in detecting compact well separated clusters*. J. Cybernetics, vol. 3, no. 3.

[Capps1992] Capps, Michael H. and Ansley, Norman, *Numerical Scoring of Polygraph Charts: What Examiners Really Do*, in: Polygraph, 1992, 21, pp. 264-320.

[Choe1992] Choe, Howon and Jordan, Jay B., *On the Optimal Choice of Parameters in a Fuzzy C-means Algorithm*. IEEE International Conference on Fuzzy Systems, San Diego, California.

[Jacobs1993] Jacobs, Eric,*Time Domain Features for The Fuzzy Set Classification of the Polygraph Data*. Master's project, Dept. of Elect. Engr., San Jose State University, California.

[Johnson1991] Johnson, Phillip E., *Darwin on Trial*. InterVarsity Press, Downers Grove.

[Jou1993] Jou, Chi-Cheng, *Supervised Learning in Fuzzy Systems: Algorithms and Computational Capabilities*. Second IEEE International Conference on Fuzzy Systems, San Francisco, California.

[IIScorp1993] Bezdek, James C., *Fuzzy Logic Inference Systems*. A Five Day Short Course, Inteligent Inference Systems Corp., San Francisco, California.

[Keller1989] Keller, J.M., Gray, M.R. and Givens J.A., *A Fuzzy K Nearest Neighbor Algorithm*. IEEE Trans. on Syst. Man. Cybernetics, vol SMC-15, no. 4.

[Layeghi1993,1] Layeghi, Shahab, *Pattern Recognition of the Polygraph Using Fuzzy Set Theory*. Master's project, Dept. of Elect. Engr., San Jose State University, California.

[Layeghi1993,2] Layeghi, Shahab, *A Comparison of Fuzzy Logic Algorithms for Pattern Recognition*. Dept. of Elect. Engr., San Jose State University, California.

[Layeghi1994] Layeghi, Shahab, *Polygraph Classification Project: A Brief Guide.* Dept. of Elect. Engr., San Jose State University, California.

[MathWorks1993] The MathWorks, Inc., *The student Edition of MATLAB*, Englewood Cliffs, NJ, Prentice Hall.

[Morris1987] Morris, Henry M., *Scientific Creationism*. Master Books, El Cajan, California.

[Olsen1983] Dale E., et. al., *Recent developments in polygraph testing*: A research review and evaluation - A technical memorandum. Washington DC, US Government Printing Office.

[Reid1966] Reid, John E. and Inbau, Fred E., *Truth and Deception: The Polygraph ("Lie Detector") Technique*.The Williams & Wilkins Company, Baltimore, Md.

[Ruspini1969] Ruspini, Enrique H., *A new approach to clustering*, Information & Control systems vol. 15 No.1.

[Wang1993] Wang, L. X., Mendel, J.M., *Fuzzy Adaptive Filters, with application to nonlinear channel equalization*. IEEE Trans. on Fuzzy Systems, 1, no. 3.

[Wang1994] Wang, L. X., *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ, Prentice Hall.

[Widrow1985] Widrow, B. and Stearns, S.D., *Adaptive Signal Processing*, Englewood Cliffs, NJ., Prentice Hall.

[Zadeh1965] Zadeh, Lotfi A., *Fuzzy sets*, Information and Control, vol. 8, pp. 338-332.

[Zadeh1975] Zadeh, Lotfi A., *Calculus of fuzzy restrictions*, in: L. A. Zadeh, K. S. Fu, K. Tanaka and M. Shimura, eds., Fuzzy Sets and Their Applications to Cognitive and Decision Processes. Academic Press, New York, pp. 1-39.

[Zadeh1970] Zadeh, Lotfi A. and Bellman, R.E., *Decision-making in a fuzzy environment*. Managment Science, 17(4).

[Zimmermann1993] Zimmermann, H.J., *Prinzipien, Werkzeuge, Potentiale*, in: FUZZY Technologien. VDI-Verlag, Düsseldorf, Germany.

Report No. DoDPI96-R-0002

Errors in the "Relevant Only" Data

San Jose State University
Department of Electrical Engineering
San Jose, CA 95106

December 19, 1995

Department of Defense Polygraph Institute
Fort McClellan, AL 36205

NON-DECEPTIVE DATA

KEY

***standard**: CODE.011, 012, 013, 021 022, 023, 031, 032, 033

****Index**: error message in MATLAB reads,

>>process
"Index exceeds matrix dimensions.

>>Error in==>c:\users\ulka\non\extractf.m
on line 48==> start = begin(i) + 30 .*times(first_channel,1);

>>Error in==>c:\users\ulka\non\process.m
on line 6==>feature = extractf(z, feature_list);"

^read3: CODE.01c, .02c, .03c, .023, .033, .011, .021, .031, .013
confusing as to how to READ3 these files

***N/A: discs were unable to be processed

^^extra: CODE.041, .042, .043 processed as t4

			NON-DECEPTIVE DATA			
	ERS	SUB #	CODE	# OF FILES	EXTRA FILES	ERRORS
1	1	2	$$EACOWO	standard*	none	none
2	1	4	$$EAD5LX	standard	none	none
3	1	6	$$EANWKF	13	0.005	none
4	1	8	$$EAOZD6	standard	none	none
5	1	9	$$EAQWB9	standard	none	none
6	1	11	$$EARKZ6	standard	none	none
7	1	12	$$EARJS0	standard	none	none
8	1	13	$$EA%KR9	standard	none	index** t3
9	1	15	$$EA%H#L	standard	none	none
10	1	18	$$EB2IYL	standard	none	none
11	1	22	$$EC4QN3	standard	none	none
12	1	26	$$EC7N7X	standard	none	none
13	1	33	$$ECLMTU	standard	none	none
14	1	34	$$ECMA%C	standard	none	none
15	1	35	$$ECM7GX	standard	none	none
16	1	36	$$ECMWB3	standard	none	none
17	1	40	$$EC#G2O	standard	none	none
18	1	43	$$EC$O0F	standard	none	none
19	1	44	$$ED8O5U	standard	none	none
20	1	45	$$ED8LUI	standard	none	none
21	1	46	$$ED9439	9	read3^	N/A***
22	1	47	$$ED9TCX	standard	none	none
23	1	50	$$EDBQR2	standard	none	none
24	1	53	$$EDCZYZ	12	extra^^	none
25	1	59	$$EDPY4#	standard	none	none
26	1	60	$$EDQCY9	standard	none	none
27	1	61	$$EDQ28X	standard	none	none
28	1	62	$$EDQOCF	standard	none	index t1
29	1	65	$$EDRKGO	standard	none	none
30	1	66	$$EDRMU#	standard	none	none
31	2	11a	$$FZIMEU	13	.005, extra	index t1a
	2	11b	$$FZISQ#	standard	none	none
32	2	12	$$FZIT4L	standard	none	none
33	2	14	$$FZJ52#	standard	none	index t1
34	2	30	$$FZZN1Y	10	0.005	index t3
35	2	32	$$FZ#D6J	10	0.005	none
36	2	33	$$FZ#0HX	13	.005, extra	div by zero t3
37	2	35	$$FZ$3A&	standard	none	none
38	2	36	$$F#8CY9	11	.005,.STR	none
39	2	38	$$F#9FJL	10	0.005	index t2, t3
40	2	41	$$F#B6SC	standard	none	none
41	2	42	$$F#B6C#	standard	none	none
42	2	45	$$F#NMDX	standard	none	index t1
43	2	47	$$F#NHQT	standard	none	none
44	2	48	$$F#&7GC	standard	none	index t3
45	2	51	$$F#QJTF	standard	none	none
46	2	52	$$F#S0KR	standard	none	none

NEWS.XLS

	ERS	SUB #	CODE	# OF FILES	EXTRA FILES	ERRORS
47	2	**53**	$$F#RRD5	standard	none	none
48	2	**54**	$$F#RYFR	12	extra	**index t3**
49	2	**55**	$$F#SALQ	10	0.005	**index t3**
50	2	**56**	$$F$C#2#	standard	none	none
51	3	**2**	$$F$D%YR	standard	none	none
52	3	**12**	$$F$I41X	11	.005,.STR	none
53	3	**25a**	$$F$IUY0	10	0.005	none
	3	**25b**	$$F$UI3X	11	.005, .STR	none
54	3	**31**	$$F$WNSF	standard	none	none
55	3	**43**	$$F%51&G	10	.STR	**index t1**
56	3	**46**	$$F%5$UF	standard	none	none
57	3	**49**	$$F%7K#0	standard	none	none
58	3	**59**	$$F%JAK6	standard	none	none

DECEPTIVE DATA

KEY

***standard**: CODE.011, 012, 013, 021 022, 023, 031, 032, 033

****Index**: error message in MATLAB reads,

```
>>process
"Index exceeds matrix dimensions.

>>Error in==>c:\users\ulka\non\extractf.m
on line 48==> start = begin(i) + 30 .*times(first_channel,1);

>>Error in==>c:\users\ulka\non\process.m
on line 6==>feature = extractf(z, feature_list);"
```

@format: files were unable to be read. Error message in DOS reads:

```
>format not linked
>abnormal program termination
```

^^extra: CODE.041, .042, .043 processed as t4

^read3: CODE.01c, .02c, .03c, .04c
confusing as to how to READ3 these files

DECEPTIVE DATA

	ERS	SUB #	CODE	# OF FILES	EXTRA FILES	ERRORS
1	1	1a	$$G3#SGD	standard*	none	index** t3a
	1	1b	$$EACLB6	standard	none	none
	1	1c	$$G3$6HN	standard	none	none
2	1	5	$$EAN#XO	standard	none	none
3	1	7	$$EAOQXV	standard	none	none
4	1	10	$$EAQ%%U	standard	none	none
5	1	14	$$EB0289	standard	none	none
6	1	16	$$EA%%MX	standard	none	none
7	1	19	$$EB2WE$	standard	none	index t3
8	1	23	$$EC4%GO	11	.005, .STR	format@
9	1	24	$$EC77GI	standard	none	none
10	1	25	$$EC76OR	standard	none	none
11	1	27	$$ECIX9#	standard	none	none
12	1	28	$$ECIVB0	standard	none	none
13	1	29	$$ECJHKO	standard	none	none
14	1	30	$$ECJVSI	standard	none	index t1, t2
15	1	31	$$ECJ#Z$	standard	none	index t3
16	1	32	$$ECLODC	standard	none	none
17	1	37	$$ECXAPG	standard	none	none
18	1	38	$$ECYCG0	standard	none	none
19	1	41	$$EC#$FA	standard	none	index t3
20	1	42	$$EC$ANC	standard	none	none
21	1	48	$$ED9$N#	standard	none	none
22	1	51	$$EDB$S3	standard	none	none
23	1	52	$$EDCSRC	standard	none	none
24	1	54	$$EDDBUX	standard	none	none
25	1	55	$$EDCBSU	standard	none	none
26	1	56	$$EDDHTI	standard	none	none
27	1	58	$$EDP26U	12	extra^^	index t1
28	1	63	$$EDQYMF	standard	none	none
29	1	64	$$EDR3XI	standard	none	none
30	1	67	$$EDS3ZL	standard	none	none
31	2	1	$$FZ3Z5S	standard	none	none
32	2	2	$$FZ3XG6	standard	none	none
33	2	5	$$FZ52G6	standard	none	none
34	2	6	$$FZ6&46	standard	none	none
35	2	8	$$FZ7B#C	standard	none	none
36	2	9	$$FZ7GP#	standard	none	none
37	2	10	$$FZIMEU	17	extra, .005, read3^	index t1
38	2	13	$$FZJ358	10	0.005	none
39	2	17	$$FZL9ZR	10	0.005	index t2
40	2	18	$$FZLBY&	standard	none	none
41	2	21	$$FZMQ#C	10	0.005	none
42	2	22	$$FZMW$H	10	0.005	index t2
43	2	25	$$FZWQQC	standard	none	index t1
44	2	26	$$FZW5T#	standard	none	none
45	2	27	$$FZYCM&	13	extra, .005	index t3

	ERS	SUB #	CODE	# OF FILES	EXTRA FILES	ERRORS
46	2	31	$$FZZR&C	12	extra	index t2
47	2	44	$$F#NC4B	standard	none	none
48	2	46	$$F#NGH3	10	0.005	none
49	2	49	$$F#&KWF	10	0.005	none
50	2	50	$$F#PUDW	standard	none	none
51	3	14	$$F$IK&0	standard	none	none
52	3	16	$$F$RJK6	standard	none	none
53	3	36	$$F%3C19	standard	none	none
54	3	40	$$F%4&C9	11	.005, .STR	none
55	3	41	$$F%4V0U	standard	none	none
56	3	54	$$F%I45#	11	.005, .STR	index t1
57	3	62	$$F%L350	standard	none	none
58	3	66	$$F%LXJ&	standard	none	none